Commodore

Cena 10 tys. zł
nr indeksu 355275

5 '92

KEDAD

Miesięcznik Użytkowników Komputerów C-64 i Amiga



Commodore



nr indeksu: 355275

Wydawca:
KEBAB - sp. z o.o.
ul.Wojciechowskiego 28
PL - 71 476 Szczecin
telefon: (091) 77674
telefax: (091) 45402

Redaguje kolegium w składzie: Krzysztof Kobus, Patryk Łogiewa, Grzegorz Mikuła, Krzysztof Moroń, Paweł Sołtysiński.

Prezes zarządu Spółki:
Piotr Sołtysiński

Szef działu AMIGA:
Krzysztof Kobus
tel.:(091)525336

Szef działu C-64:
Paweł Sołtysiński
tel.:(091)77674

Stale obecni na łamach:
Robert Turliński
Miłosław Smyk
Arkadiusz Zych

Redakcja nie zwraca nie zamówionych materiałów, oraz zastrzega sobie prawo wprowadzania zmian w otrzymanych rękopisach.

Projekt okładki:
Dariusz Zawadzki

Zdjęcia:
Sławomir Borek /"Panorama"

W sprawie kolportażu...

I TY MOŻESZ BYĆ KOLPORTEREM "KEBABA"!

Chcielibyśmy zainteresować naszą propozycją te osoby fizyczne i prawne, które chciałyby się podjąć kolportażu we własnym zakresie. Zapewnimy jednocześnie co najmniej tygodniowe wyprzedzenie przed przekazaniem nakładu do dystrybucji przez krajowego kolportera. Zapraszamy do współpracy studia komputerowe, księgarnie i obrotne osoby indywidualne. Szczegółowych informacji idziela się telefonicznie i w siedzibie redakcji.

... reklamy ...

Przedsiębiorstwo KEBAB spółka z o.o. oferuje Państwu szybką i tanią obsługę reklamową. Ogłoszenia drobne od osób indywidualnych (do 10 słów na wyciętym z numeru kuponie) przyjmujemy bezpłatnie. Większe - 1000 zł za słowo. Reklamy ramkowe (minimalny format - 20 cm kwadratowych): 1 cm2 - 4500 zł; cała strona - 2,5 miliona zł; dodatkowy kolor - odpowiednio 50% drożej. Treść ogłoszeń przyjmujemy za pośrednictwem poczty (adres - patrz stopka redakcyjna) lub Agencji Informacyjnej "SIEĆ"; Warszawa, ul.Prezydencka 11; tel.255-433; fax.254-164. Ogłoszenia wraz z określeniem formatu reklamy prosimy nadsyłać listem poleconym. Dołączenie odcinka wpłaty znacznie przyspieszy zamieszczenie ogłoszenia.

... i prenumeraty.

Aby uporać się z problemem ciągłego wzrostu cen usług poligraficznych, papieru itp. i uniknąć dokonywania przez Czytelników Kłopotliwych dopłat, postanowiliśmy wprowadzić tzw. małą prenumeratę w okresach 3-miesięcznych. Rozwiązanie to gwarantuje Czytelnikom niezmienność ceny w okresie, który obejmuje zamówienie. Cenę egzemplarza wraz z kosztem usługi pocztowej skalkulowaliśmy na 9500 zł. Daje to następujące możliwości:

numery 6, 7, 8 - 28500,-

KEBAB sp. z o.o.
Pomorski Bank Kredytowy II Oddział w Szczecinie
konto nr: 368113-25771-136

Należy również podać DOKŁADNY ADRES, IMIĘ I NAZWISKO zamawiającego!



Nr 5 maj 1992

AMIGA 500 - PC
Test najpopularniejszych emulatorów IBMa - czytajcie od strony 18

ARexx
Jeśli chcesz aby to hasło przestało być tajemnicze, zerknij na stronę 5

Assembler na C-64
Czwarty odcinek kursu - strona 7

Kupiłem C-64...
... i uczę się Basic'a.
Od strony 16

Action Replay na C-64
Czego konstruktorzy nie przewidzieli.
Czytaj na stronie 11

Lubisz pograć ?
Czytaj od strony 24

Spis Treści:

...NEWS...NEWS...NEWS...NEWS...

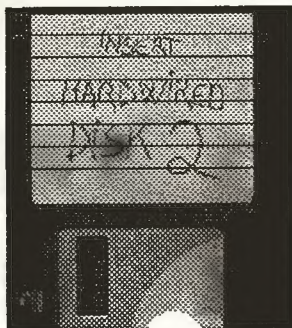
Właścicieli instalacji do odbioru telewizji satelitarnej ucieszy z pewnością informacja o planowanym przez MTV wyemitowaniu video-dema grupy The Silents zatytułowanego "Global Trash II". Ta najpopularniejsza wśród europejskiej młodzieży stacja rozrywkowa, w ciągu zaledwie kilku tygodni od premiery dema (teledysku?) otrzymała ponad tysiąc listów z prośbami o jego prezentację. Kto wie, może dokona jej Ray Cokes w "THE MOST WANTED"? Natomiast grupa Silents nie zamierza spocząć na laurach i już zapowiada kolejny videoclip zrealizowany przy pomocy Amigi i techniki wideo - "Hardwired II".

4-Mat, jeszcze do niedawna zadowolony pracownik Sonic Projects (szczegóły -patrz Kebab 2-3/92) postanowił na razie zrezygnować z zawodowstwa i jako członek grupy Anarchy powrócić na łono sceny, której jak sam twierdzi, bardzo mu brakuje.

Inną znaną postacią, której sygnatura od dłuższego czasu nie pojawiła się pod żadną nową produkcją jest Romeo Knight/ TRSI. Ten utalentowany muzyk, twórca między innymi podkładu dźwiękowego do bardzo w swoim czasie popularnego dema "Cebit'90", zasugerował możliwość powrotu na scenę. Póki co, wspólnie z członkiem grupy D-TECT posługującym się pseudonimem Unknown, kontynuuje pracę nad nową grą.

Miłośnicy perfekcyjnego kodu, doskonałej muzyki, jeszcze lepszej grafiki i tym podobnych ochów i achów mogą zacząć ostrzyć sobie zęby na kolejną, trzecią część legendarnej już gry "Shadow of the Beast". Firma Psygnosis zapowiedziała ostateczną premierę tego obiecywanego już od dawna, lecz wciąż spóźniającego się programu na lato bieżącego roku.

Jest już natomiast na rynku "Project X" - gra typu shoot'em up, napisana przez znany z wysokiej jakości produktów zespół programistów Team 17. Doskonała muzyka - Allister Brimble, Bjorn A. Lynne (ex Dr.-Awesome), szybka animacja (co 1/25 sek.) na ekranie w overscan'ie, niesamowita ilość grafiki w tle i zapierająca dech w piersiach liczba kolorów to tylko niektóre atrybuty tej czterodyskowej gry.



HARDWIRED

Walka firm software'owych z piratami, mimo wciąż nowych pomysłów na zabezpieczanie programów wydaje się równie skuteczna jak opcja OPTIMIZE w starszych wersjach X-COPY. Przekonała się o tym niedawno firma OCEAN, a dokładniej jej dyrektor do spraw oprogramowania - Gary Bracey. On to bowiem był najgorętszym zwolennikiem wyposażenia jednego z nowych produktów firmy, gry "Robocop II" w sprzętową przystawkę - tzw. keylock, mającą o ile nie zapobiec nielegalnemu rozpowszechnianiu kopii, to przynajmniej opóźnić je na tyle, by OCEAN zdążył wyciągnąć korzyści ze sprzedaży oryginału. Sam keylock był określany jako kolejna generacja znanych z krainy IBM "dongli", urządzenie o wysokim stopniu komplikacji, prowadzące przez cały czas pracy swoisty "dialog" z komputerem - którego zatem podrobienie w warunkach domowych byłoby tak pracochłonne i kosztowne, iż po prostu nieopłacalne. Firma, aby nie obciążać kieszeni nabywców, postanowiła nawet nie doliczać wartości przystawki do ceny programu, licząc na to, że dodatkowe koszty zwrócą się na skutek braku pirackich kopii. Zachodnie magazyny rozpisywały się również na temat nowego, rewelacyjnego, pięćdziesięciopoziomowego(!) software'owego systemu zabezpieczającego, przy czym warto wspomnieć, że niezwiązany z projektem programista firmy OCEAN potrzebował tygodnia(!!) na "połamanie" tylko jednego spośród owych poziomów. Jakże wielkie musiało być rozczarowanie szefów, gdy "Robocop II" odbezpieczony uprzednio w przeciągu kilku dni (niektóre źródła mówią tu nawet o jednej nocy), działający wyśmienicie bez keylock'a, został w błyskawicznym tempie rozprowadzony po świecie przez, sławnych już z innych tego typu wyczynów, członków grupy Fairlight.

Miłosław "Thorgal" Smyk

Wyjaśniła się już sprawa nieobecności firmy Commodore na serii targów AmiShows. Po prostu ze względu na "pewne nieporozumienia" firma Commodore zrezygnowała ze współpracy z firmą AmiShows Europe. Efektem tego jest między innymi powołanie do życia przez CBM własnej imprezy targowej "World of Commodore" we Frankfurcie nad Menem.

Coraz więcej firm decyduje się na produkcję własnych kart przyspieszających do Amigi opartych na procesorze 68040. G-Force 040, Fusion forty czy Turbo 40 to tylko najbardziej znane. Amiga wyposażona w taką kartę z zegarem 28 MHz osiąga szybkość ponad 20 MIPS (milionów instrukcji na sekundę) dla porównania najszybsze komputery wyposażone w procesor 80486 osiągają "zaledwie" około 16 MIPS.

Coraz więcej również mamy nowych napędów przeznaczonych do odczytu CD-ROM'ów. Znane firmy jak NEC czy Toshiba dołączyły ostatnio do producentów tego typu urządzeń dla Amigi. Również firma Commodore produkuje już napędy przeznaczone dla A500 i podłączane na wzór twardego dysku A590, z boku obudowy.

Firma Gold Vision oferuje zestaw 151 obiektów wektorowych przeznaczonych do zastosowań DTP. Grafiki podzielone są na jedenaście grup a w skład pakietu kosztującego ca. 100,-DM wchodzi również program umożliwiający konwersję do formatów Aegis-Draw, EPS, IFF, GEM Metafile i ProDraw Clip.

SDI

Na ostatnim copy-party w Warszawie wypuszczono kilka ciekawych programów demonstracyjnych. Część z nich postaram się Wam przybliżyć.

MORE THAN NOPS / TABOO

Jest to pierwsze demo tej nowej grupy z Katowic. Składa się ono z intra oraz 5 części. Demo zajmuje całą stronę dysku i ładuje się bezpośrednio po sektorach (z pominięciem katalogu) tak, że niemożliwe jest wczytanie poszczególnych części oddzielnie. Grafika nie jest najmocniejszą stroną tego programu, ale spowodowane jest to tym, że grupa ta nie posiada grafików i koderzy sami musieli tworzyć oprawę plastyczną. Kod jest na dobrym poziomie, ale same efekty byłyby może nużące (dominują sidebordery) gdyby nie towarzyszyły im ciekawe pomysły. Należy do nich m.in. "gra telewizyjna", połączona z digitalizowaną muzyką, gra w "bzzz.." po bocznych ramkach czy ładny (choć miejscami dość wolny) zoomer. Muzyka jest różnych autorów, wyciągana z innych dem. Jako debiut demo to wypadło bardzo dobrze.

A PIECE OF SHIT / PARADOS

Kolejne demo chłopców z Parados'u, tym razem chyba nieco słabsze od poprzedniego. Demo składa się z 6 części oraz intra. Uwagę zwraca własna muzyka (autorstwa JADOONE'a), co jest żadkością w Polsce. Grafika jest na średnim poziomie ale na pewno lepsza niż w poprzednim demie. Każdej części towarzyszy obrazek u dołu ekranu. Dużą wadą tego dema jest jego monotonność. Większość części wygląda bardzo podobnie i różni się tylko efektem w środkowej części ekranu. Powoduje to lekkie znudzenie podczas oglądania. Kod stoi na wysokim poziomie, chociaż obok nowych efektów znajdują się starsze, znane już z dem innych grup. Niemniej jednak tej stronie dema prawie nic zarzucić nie można.

Demo to, mimo że nie najlepsze, pozostaje ciągle całkiem przyzwoitym jak na polskie warunki.

LAST WORK 4 ASX / ASPHYXIA

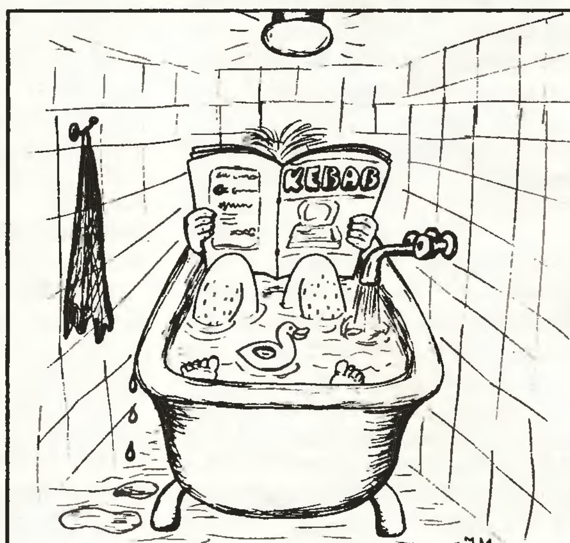
Ostatnie demo, które mam zamiar opisać jest (jak głosi nazwa) pożegnalnym demem Rekina/Asphyxia, który po jego wydaniu przeszedł do Parados'u. Składa się ono tylko z 2 części poprzedzonych intrem. Tym co uderza w tym demie jest świetna grafika Carriona (obecnie w grupie Elysium): oryginalne pomysły i dobre wykonanie. We wszysykich częściach jest użyty ten sam efekt (V.S.P.) za każdym razem jednak odmiennie więc nie powoduje to znużenia. Kod sprawia raczej wrażenie podporządkowanego grafice i służącego jedynie lepszemu jej zaprezentowaniu. Muzyka jest wyciągnięta z innych dem, niemniej jednak wybrano muzyczki ładne i pasujące do

(c.d. strona 6)

Dlaczego wolę Amigę ? - po raz drugi

Przyglądając się zaistniałej na łamach polskich czasopism komputerowych dyskusji na temat "Dlaczego wolę..." postanowiłem również dołączyć kilka swoich uwag do wcześniej już wypowiedzianych bądź to w "Bajtku" bądź w "KEBAB'ie". Ponieważ nie chciałbym aby moja wypowiedź przypominała żywo czasy rozmów na temat co jest lepsze (Spectrum czy Atari 800), napiszę po prostu: Wolę Amigę bo jest komputerem DOMOWYM. Co za tym idzie wymagania stawiane przeze mnie takiemu komputerowi są wyższe niż te, które musi spełniać komputer OSOBISTY. Oprócz wszystkich żmudnych zadań typu edycja tekstów, prowadzenie baz danych, DTP, wykonywanie obliczeń przy użyciu arkuszy kalkulacyjnych bądź rysunków mniej lub bardziej technicznych (CAD), Musi być on również źródłem rozrywki i to niekoniecznie tej polegającej na "zjadaniu, strzelaniu i udeptywaniu" choć nie przeczę że jest to najpopularniejszy sposób dostarczenia rozrywki np. własnym pociechom przy pomocy komputera. Ten cel najprawdopodobniej przyświecał twórcom Amigi jako komputera (początkowo miała to być tylko konsola do gier), którzy zauważywszy kryjące się w tej konstrukcji możliwości postanowili, że właśnie Amiga powinna zastąpić na rynku komputerów domowych wszystkie królujące tam wtenczas konstrukcje. Tak również należy chyba rozumieć cytowaną już w KEBAB'ie wypowiedź szefa "Electronic Arts". Nauczono się już jednak, że komputer który nie posiada tzw. otwartej architektury i co za tym idzie możliwości ciągłego rozwoju nie ma szans na utrzymanie się na rynku. No chyba, że jest to C-64 ale to wyjątek od reguły. Czy to się sprawdza w praktyce? Wystarczy się rozejrzeć po sklepach komputerowych w Europie. Na półkach oprócz wyrobów będących kolejnymi "klonami" komputerów typu IBM PC. Można znaleźć jeszcze czasami Atari ST/TT. Czasami wspomniany wyżej C-64 no i Amigę począwszy od najprostszych "pięćsetek" dostępnych w każdym domu towarowym a na Amidze 3000 kończąc. Za oceanem sprawa ma się nieco inaczej. Tam króluje firma Apple i jej kolejne wersje Macintosh'a. Amiga natomiast jest podobną egzotyką jak w Europie Macintosh. Ale tak jak u nas nie krytykuje się firmy Apple za to, że ich produkty są mało rozpowszechnione na naszym kontynencie, tak również w Ameryce nie podlega z tych przyczyn krytyce firma Commodore-Amiga. Jednakże europejski sukces rynkowy Amigi nie był wcale łatwy. Promocja każdego nowego standardu komputerowego nie jest łatwa. Aby komputer dobrze się sprzedawał musi istnieć na rynku odpowiednia baza oprogramowania jak również tzw. urządzeń peryferyjnych, chociażby joystick'ów (patrz przykład C-16/116/plus4). Aby jednak firmy produkujące oprogramowanie czy peryferia zainteresowały się produkcją przeznaczoną dla określonego modelu komputera muszą mieć zapewniony zbyt czyli musi wcześniej zostać sprzedana taka ilość egzemplarzy tego samego komputera aby potencjalny producent np. firma software'owa mógł bez obaw zainwestować w produkcję dlań przeznaczoną. Tu kółko się zamyka! Zanim zatem zaczniemy krytykować Amigę przeciwstawiając jej doskonałe możliwości oprogramowania na komputerach klasy PC (lub co gorsza możliwości sprzętowe), należy zwrócić uwagę na to, że Amiga ustaliła swoją pozycję na rynku nie dzięki oprogramowaniu bogatych firm typu MicroSoft, Borland itp. lecz głównie dzięki wysiłkom domorosłych programistów oraz Hacker'ów piszących w przerwie między szkołą a snem doskonałe programy demonstracyjne oraz gry. Tak jak poprzez długą drogę ewolucyjną programy użytkowe na komputery PC osiągnęły już dzisiaj wysoki poziom, tak z tych samych przyczyn gry oraz demonstracje pisane na Amigę są często "zapierające dech w piersiach". Podobnie rzecz się ma także z C-64 i również z tych samych powodów. Lecz na dzień dzisiejszy nie jest już prawdą to, że nie istnieje na Amigę oprogramowanie spełniające wymogi profesjonalistów. "Grube Ryby" na rynku software'owym zwęszyły w końcu interes w pisaniu dobrych programów i całych pakietów na ten komputer. Co więcej wielu byłych Hacker'ów założyło własne firmy i ugruntowało

GYROS!



sobie pozycję właśnie dzięki Amidze. Nieprawdą również byłoby gdybym powiedział, że nie ma dobrych gier na IBMach. Również i tu mamy ostatnio duży ruch. Wprowadzenie kart graficznych typu VGA otwarło drogę do powstania gier na bardzo wysokim poziomie graficzno-animacyjnym. Pozostają co prawda jeszcze problemy typu joystick czy dźwięk ale i z tym można sobie poradzić. Powstały już karty dźwiękowe dzięki którym można w końcu usłyszeć coś przyjemniejszego dla ucha niż brzęczenie przypominające iskrzenie uszkodzonego monitora. Czy zatem faktycznie porównywanie PC z Amigą "nie ma sensu"? Na pewno takie porównanie jak uczynił pan Dybowski w swoim artykule który z resztą stał się inspiracją do moich (i nie tylko moich) wywodów nie prowadzi do żadnych sensownych i "pozbawionych emocji" wniosków. Moja sugestia na którą czeka autor wyżej wymienionego artykułu byłaby: Porównać standardową Amigę A500 do komputera typu IBM PC/XT turbo (zbliżona częstotliwość pracy zegara taktującego) bez twardego dysku za to z jednym tylko napędem dysków elastycznych (bardzo zbliżona konfiguracja) i z 256 KB pamięci RAM (podobna ilość wolnej pamięci pozostającej do dyspozycji użytkownika po pełnym uruchomieniu systemu operacyjnego). Porównując ewentualnie systemy operacyjne, należy uwzględnić całość obydwu systemów, pamiętając o tym, że w Amidze ogromna jego część mieści się na dysku i to nie bez logicznej przyczyny. Życząc ciekawych efektów testowych łączę pozdrowienia dla wszystkich fanów komputerów choćby obiektem ich uwielbienia był np. ZX81 firmy Sinclair.

Zenon Majchrzak

64 Cartridge ? Czy nie Cartridge ? - Black Box na C-64

Dzisiaj zajmiemy się rodziną cartridge'ów sprzedawanych na polskim rynku pod wspólną nazwą "Black Box". Jest to chyba jedyny cartridge dostępny na polskim rynku, którego zawartość została połączona w całość (proszę zwrócić uwagę, że nie używam tu słowa "zaprogramowana") w Polsce. Spośród całej plejady czysto pirackich produktów sprzedawanych przez tego samego producenta jest to bardzo sympatyczny akcent choć nie do końca jest prawdą to, że firma jest właścicielem praw autorskich do programów zawartych wewnątrz cartridge'a. O ile większość procedur pochodzi z tzw. Public Domain (np. Turbo Tape) o tyle właścicielem korektora skosu głowicy magnetofonowej jest niemiecka firma Heinz Heisse Verlag. Nie przeszkadza to bynajmniej w tym aby znajdował się on w "polskim" cartridge'u. Co prawda producent w instrukcji do innego swojego cartridge'a przyznaje w końcu, że program ten jest cytuję "zmienioną i udoskonaloną wersją niemieckiego programu RECORDER JUSTAGE...". Jednakże przynajmniej w przypadku Black Box'a wprowadzone "udoskonalenia" polegają na usunięciu komunikatów producenta oraz zmianie nazwy programu. Zainteresowanym przytoczę drobny fakt - wyjaśnienie. Po uruchomieniu korektora komendą HF zgłasza się menu z którego możemy wybrać F1 - normal tape oraz F3 - Turbo Tape. W przypadku poprawnego ustawienia głowicy (jak najwęższe prążki) pojawiający się wykres powinien zmieścić się pomiędzy liniami, które przywołaliśmy na ekran wciśnięciem odpowiedniego klawisza. O ile standardowa transmisja (normal tape) pozwala na swobodne ustawienie prążków pomiędzy liniami pomocniczymi o tyle transmisja Turbo za nic nie chce się zamknąć pomiędzy przewidzianymi ponoć dla niej liniami. Nie przeszkadza to zbytnio w korzystaniu z programu gdyż wystarcza takie ustawienie głowicy aby prążki wykresu były najwęższe. Niemniej jednak zastanawia fakt po co autorzy stworzyli w programie linie pomocnicze dla Turbo Tape, które w niczym nie pomagają? Niewiele osób dzisiaj wie, że oprócz systemu Turbo napisanego po raz pierwszy chyba w roku 1984 przez Stefana Senz'a, pojawił się w roku 1985 lansowany przez wymienioną już firmę system nazwany "SuperTape". System ten oprócz tego, że był jeszcze szybszy od Turbo Tape pozwalał m. in. na używanie w nazwach zbiorów tzw. "jokerów" tj znaków zastępczych (?, *) podobnie jak w przypadku stacji dysków. Umożliwiał również obsługę zbiorów sekwencyjnych (SEQ) itd. Ponieważ firma wszystkie swoje produkty sprzedawała na kasetach zapisanych w tym systemie to postanowiła również wyprodukować program do korekcji skosu głowicy (im większa szybkość tym większa wrażliwość na niewłaściwe ustawienie głowicy). Tak! Jest to ten sam program. I jeżeli mamy gdzieś kasetę zapisaną w systemie SuperTape np. któreś wydanie INPUT 64 to możemy sprawdzić, że istotnie przy poprawnym ustawieniu prążki trafiają dokładnie pomiędzy pojawiające się linie. No dobrze zostawmy już problemy piractwa (zamierzonego lub nie) na boku i przejdźmy do pozostałych procedur zawartych wewnątrz cartridge'a. Firma wypuściła na rynek kilka wersji "Black Box'a" z których mnie osobiście najbardziej przypadły do gustu dwie pierwsze. Jeżeli nie liczyć kilku drobiazgów np. typu wadliwa konstrukcja płytki drukowanej cartridge'a powodująca niezgodność z wszystkimi wersjami płyty głównej C-64 za wyjątkiem dwóch ostatnich (było ich dotychczas sześć), konstrukcja obudowy (patrz KEBAB 1/92) oraz kilka "niedopatrzeń" odnośnie praw autorskich. to trzeba przyznać, że dobór procedur (przynajmniej do wersji drugiej włącznie) jest dosyć udany i użytkownik Commodore 64 z magnetofonem może mieć z takiego zestawu narzędzi sporo korzyści. Przestrzec natomiast należy użytkowników starszych wersji

C-64. Na rynku znajdują się już pirackie kopie (pirackich częściowo) cartridge'ów Black Box V2 i co ciekawe są one skonstruowane w taki sposób, że działają (!) również na starszych modelach "sześćdziesiątki czwórki". Przestrzec również należy potencjalnych nabywców wersji V3 i V4. Zwłaszcza ostatnia z nich posiada tyle "ulepszeń", że duża ilość programów po prostu nie ma ochoty działać dopóki nie wyjmiemy cartridge'a. Dopóki jednak nie pojawi się w pełni polska alternatywa dla Black Box'a o której krążą ostatnio plotki (ma się nazywać Turbo Tower) pozostaje on mimo swoich wad najtańszym zestawem narzędzi ułatwiających życie użytkownikom Datasette dla C-64.

SD!

ARexx

Królewski język

Drogi Czytelniku! Muszę z przyjemnością stwierdzić, że zasługujesz na wiele uznania. Sięgając bowiem po ten artykuł dowiodłeś, iż jesteś prawdziwym użytkownikiem Amigi - nie zajmujesz się wyłącznie graniem, ale i nieco poważniejszymi rzeczami. Amiga nadaje się bowiem (wbrew rozsiewanym przez niektórych plotkom) i do profesjonalnej pracy. Chciałbym w swym artykule przedstawić jedno z najwspanialszych narzędzi, jakie posiada nasz komputer: coś, czego nie posiada ani Atari ST ani Macintosh ani system "Windows" - język Rexx. Być może w tym miejscu słyszysz o nim po raz pierwszy. Nie ma w tym nic dziwnego - nie jest to język tak powszechnie używany jak Pascal czy C. Nie znaczy to wcale, że jest gorszy - służy po prostu do czego innego.

Trudno mi powiedzieć, kiedy Rexx został zdefiniowany. Według brytyjskiego czasopisma "Amiga Computing" było to w roku 1985. Języka tego użył (uwaga!) koncern IBM (tak, to ten sam) jako standardowego dla wszystkich produkowanych przez tę firmę komputer"w, począwszy od tzw. "mainframes", poprzez minikomputery a skończywszy na zwykłych PC działających w systemie OS/2. Amigowska wersja nosząca nazwę ARexx została zaimplementowana przez Williama S. Hawesa w roku 1987 i jest kompatybilna z wersją IBM. Ta jego cecha jest czasami wykorzystywana - na przykład Stanford University Linear Accelarator Center w USA używa Amig jako terminali dla dużych komputerów IBM właśnie dzięki istnieniu ARexx'a. Do czego można używać ARexx'a? Widzę przynajmniej 3 zastosowania:

* ze względu na stosunkowo "wysoki poziom" tego języka - do pisania samodzielnych programów o rozmaitym przeznaczeniu (w Rexx'ie napisano np. jeden z pierwszych wirusów komputerowych, tzw. "Choinkę z Clausthal" - patrz "Komputer" 11/88);

* ze względu na istnienie łatwych w zastosowaniu łączników z DOS-em - do tworzenia tzw. "batch'ów", czyli zbiorów poleceń dla DOS-u;

* ze względu na możliwość koumnikacji z innymi programami - do kreowania makro-procedur dla różnych programów użytkowych, z których na co dzień korzystamy.

Ponieważ ta trzecia możliwość wydaje się najbardziej frapująca, wokół niej zamierzam się skoncentrować. Programów współpracujących z ARexx'em jest wiele i ich liczba stale rośnie:

* Edytory tekstu: CygnusEd, ProWrite, QuickWrite, TxED+, TurboText;

* Bazy danych: SuperBase, MicroFiche Filer+;

* DTP: AmigaTex;

* Arkusze kalkulacyjne: Advantage, Professional Calc;

* Muzyczne: Bars&Pipes Pro;

* Graficzne: Digi Paint 3, Macro Paint, TAD Pro, ProVector, DesignWorks;



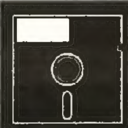
* Programowanie: CAPE 68K, CanDo;

* CAD: DynaCADD;

* Komunikacja: Baud Bandit, A-Talk III;

* Zarządzanie zbiorami: Directory Opus 3, Disk Master II;

- to lista tylko tych bardziej znanych.



Dosyć gadania, czas przejść do programowania. Aby korzystać z ARexx'a, trzeba go najpierw mieć. Mamy tutaj właściwie trzy opcje:

- skopiowanie (czytaj: ukradzenie) dyskietki z ARexx'em;

- zainstalowanie systemu operacyjnego 2.0 we własnej Amidze (Arexx jest począwszy od tej wersji systemu operacyjnego zawarty jako jego część składowa).



- kupienie Amigi wyposażonej w OS-2.0 (500plus, 600, 3000). W tym wypadku teoretycznie powinniśmy otrzymać także podręcznik, ale niestety (przynajmniej w wersji niemieckiej) nie ma go.

Niezbędny jest też CygnusEd, ponieważ dla tego programu będziemy tworzyć nasze procedury. W celu rozpoczęcia pracy musimy upewnić się, że:

- w katalogu C: znajdują się zbiory: Tco, Tcc, Ts, Te, Rx, Rxc, RexxMast;

- w katalogu LIBS: znajdują się: rexxsyslib.library i rexxsupport.library. Właściwy interpreter ARexx'a to biblioteka rexxsyslib.library (zajmuje ona około 33 KB).

Programy w Rexx'ie to zwyczajne pliki tekstowe które tworzymy dowolnym edytorem (takim jak CygnusEd czy ED). Uruchomić program możemy za pomocą jednego z następujących sposobów:

- z poziomu CLI pisząc:

Rx <RexxProgram> <parametry>;

- z poziomu Workbench'a, ustawiając za pomocą fukcji Info DefaultTool dla ikony naszego programu

na "crx";

- z programu użytkowego; np. w CygnusEd wybieramy opcję "SpecialDos/Arexx interface:Install Dos/ARexx command..." i wpisujemy najpierw numer klawisza funkcyjnego, któremu przypisujemy komendę a następnie nazwę pliku, w którym znajduje się program w ARexx'ie (np. "RAM:Prog.rexx", jeżeli program nosi nazwę "Prog.rexx" i znajduje się w ramdysku). Odtąd aby uruchomić nasze dzieło wystarczy nacisnąć odpowiedni klawisz funkcyjny.

Aha, przed korzystaniem z ARexx'a należy jeszcze uruchomić program RexxMast, który między innymi ładuje stosowną bibliotekę. Najlepiej zrobić to poprzez umieszczenie w zbiorze "Startup-Sequence" linii "RexxMast".

Pierwszy program.

Nasz pierwszy program (patrz listing 1) jest bardzo prosty. Jego jedyne zadanie to zapoznanie użytkownika Cygnusa z aktualnym czasem. Przypatrzmy mu się dokładnie.

Każdy program w ARexx'ie musi zaczynać się parą znaków: "/*". Ponieważ znaki te to jednocześnie początek komentarza (jego koniec jest oznaczany przez "*/"), więc do dobrego tonu należy umieszczanie na początku programu informacji dotyczących jego przeznaczenia, autora, itp.

W pierwszej linii po komentarzu mamy deklarację zmiennej o nazwie LF (skrót angielskiego Line Feed, oznaczającego po prostu przejście do następnej linii) i przypisanie jej wartości za pomocą znaku "=". Zapis '0A'X oznacza znak, którego kod szesnastkowy wynosi 0A. Litera "X" oznacza w tym przypadku właśnie liczbę szesnastkową. Możemy też używać liczb binarnych, zakończonych literą "B". Wspomnianą linię moglibyśmy zapisać więc następująco:

LF='1010'B.

Linia kolejna zawiera instrukcję ADDRESS. Ma ona format:

ADDRESS <NazwaPortu>

i mówi ARexx'owi, z jakim programem ma się komunikować. W tym przypadku "rexx_ced" (należy zwrócić uwagę, iż stosowane są małe litery) jest nazwą dostarczoną przez Cygnusa - a więc ARexx będzie "rozmawiał" właśnie z tym programem.

W następnej linii mamy przypisanie zmiennej Time aktualnej godziny odczytanej przez funkcję TIME('H') i minuty odczytanej przez funkcję TIME('M'). Ponieważ TIME('M') odczytuje aktualną minutę począwszy od początku dnia, więc aby uzyskać "prawdziwą" minutę musimy uzyskać resztę z dzielenia tej wartości przez 60. Dokonuje tego operator "modulo" - podwójna ukośna kreska.

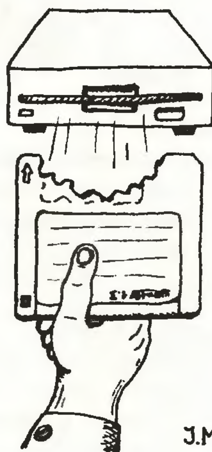
Aby przy drukowaniu rozdzielić godziny od minut używamy znaku dwukropka, który łączymy z godziną i minutą za pomocą podwójnej pionowej kreski. Tak właśnie w Rexx'ie odbywa się łączenie tekstów. Na marginesie wspomnę, że funkcja TIME() ma jeszcze kilka innych, ciekawych argumentów. Na przykład TIME('R') zeruje stoper, TIME('E') odczytuje czas ze stopera.

W kolejnej linii aktualny czas, wycentrowany w polu o szerokości 19 znaków za pomocą funkcji CENTER() zostaje dołączony na końcu tekstu "The current time is" i umieszczony w nowej linii. Wreszcie ostatnia instrukcja OKAY1 <tekst>, nie będąca instrukcją ARexx'a, lecz Cygnusa powoduje wyświetlenie na ekranie Cygnusa okienka zawierającego tekst, reprezentowany przez zmienną tekstową "TimeString". Gdybyśmy chcieli wypisać dany tekst w okienku CLI powinniśmy użyć instrukcji ARexx'a SAY <tekst>. A co zrobić, by "wpisać" tekst z ARexx'a do edytora? Należy użyć komendy Cygnusa TEXT <tekst>. Polecam eksperymentowanie z tymi instrukcjami w naszym przykładowym programie, równocześnie zapraszając do lektury za miesiąc, gdzie zaprezentuję kolejne rozkazy i przykładowe procedury.

Michał Łętowski

GYROS!





...NEWS...
(c.d. ze strony 2)

części. Mankamentem dema jest częste "zawieszanie" się komputera w czasie przechodzenia z pierwszej części do drugiej.

I to już wszystko tym razem. Jeżeli chcecie przeczytać w naszym magazynie recenzje swoich dem (programów użytkowych, grafiki czy muzyki) to po prostu przyślijcie je do mnie.

Krzysztof "BRUSH" Dąbrowski
ul.Witosa 6/4
06-300 Przasnysz

P.S. Przepraszam grupę SKYLIGHT za brak opisu ich dema, ale mój dysk uległ uszkodzeniu.

(c.d. strona 17)

Assembler na C-64

(odcinek 4)

W poprzednim odcinku nauczyliśmy się pewnych operacji na pamięci ekranu. Wypełnialiśmy ją znakiem spacji, umieszczaliśmy tam napisy itp. Tym razem podejdziemy do tego tematu z nieco innej strony. Posłużymy się procedurami systemu operacyjnego. Zanim jednak się do tego zabierzemy musimy jeszcze zaznajomić się odrobinę z konstrukcją tegoż systemu. W komputerach C-64, w przeciwieństwie do wielu innych, znajduje on się w całości w pamięci ROM. Oznacza to, że natychmiast po włączeniu zasilania system operacyjny przejmuje kontrolę nad komputerem. Skraca się w ten sposób znacznie czas uruchamiania systemu. Sama pamięć ROM podzielona jest na trzy części znajdujące się w różnych fragmentach obszaru adresowego (patrz również schemat z odcinka drugiego - KEBAB 2-3/92).

```
BASIC ROM - $A000-$BFFF
CHARACTER ROM - $D000-$DFFF
KERNAL ROM - $E000-$FFFF
```

Co prawda nazwy wskazują dosyć jednoznacznie zawartość poszczególnych fragmentów ROM'u (patrz słowniczek) niemniej jednak w rzeczywistości sprawa ma się nieco inaczej. Interpreter BASIC'a wcale nie kończy się pod adresem $BFFF lecz jego fragment (procedura obliczająca funkcję EXP()) znajduje się jeszcze w układzie oznaczonym jako KERNAL ROM. Oprócz tego interpreter jest związany wieloma innymi procedurami ściśle z systemem operacyjnym. Na końcu obszaru adresowego znajduje się tzw."KERNAL Jump Table" a po polsku tabela skoków do procedur systemu operacyjnego. Została ona umieszczona tam aby zapewnić kompatybilność oprogramowania w przypadku wprowadzenia przez producenta innej wersji ROM'u (np. z poprawionymi błędami) co zresztą miało później miejsce. Na czym to polega? Już tłumaczę! W pamięci ROM (BASIC i KERNAL) zostało umieszczonych bardzo dużo procedur niezbędnych do tego aby po włączeniu zasilania komputer można było w ogóle obsługiwać. To, że na ekranie pojawia się dobrze nam znajomy napis "**** COMMODORE 64..." nie jest zasługą samego komputera lecz właśnie procedur ROM'u.

Również ich sprawką jest to, że na ekranie miga kursor a po naciśnięciu jakiegoś klawisza pojawia się np. litera. Gdybyśmy chcieli powiedzieć co robią one od początku (tj. gdy włączamy komputer) to musiało by to wyglądać (w dużym skrócie) mniej więcej tak:

- przygotować do pracy odpowiednie obszary pamięci RAM

- przygotować do pracy wszystkie porty I/O (wejścia/wyjścia)

- przygotować do pracy procesor wizyjny (VIC)

- przygotować do pracy procesor dźwiękowy (SID)

- sprawdzić czy nie został zainstalowany samostartujący cartridge

- jeżeli tak to przekazać kontrolę do cartridge'a

- jeżeli nie to przygotować do pracy BASIC i przekazać mu kontrolę po wykonaniu tych wszystkich mozolnych czynności system operacyjny wcale nie zaczyna odpoczywać. Dopiero teraz zaczyna się dla niego prawdziwa harówka:

- 50 razy na sekundę sprawdzić czy coś się nie dzieje

A co może się dziać? Sporo! Np. może upłynąć czas po którym należy znowu "mrugnąć" kursorem. Może ktoś nacisnąć jakiś klawisz na klawiaturze. Może trzeba już zmienić wartość zmiennych systemowych TIME (TIME$). Może ktoś nacisnąć klawisz np. przewijania w magnetofonie i wypadałoby mu włączyć silnik aby mógł tą kasetę przewinąć (tak, tak! to, że po wciśnięciu klawisza w magnetofonie uruchamia się jego silnik to też zasługa systemu operacyjnego). itd.



- jeżeli coś się dzieje, podjąć odpowiednią akcję np. jeżeli został wciśnięty klawisz na klawiaturze to należy sprawdzić który to klawisz, czy tylko jeden (mógłby być np. z SHIFT'em), jaką akcję należy wykonać dla takiego klawisza (lub kombinacji klawiszy). Jeżeli jest to litera np. "A" to należy pobrać odpowiadający tej literze kod ekranowy ($01), sprawdzić w którym miejscu ekranu znajduje się kursor, przeliczyć to na odpowiedni adres pamięci ekranu i umieścić pod tym adresem pobrany kod (podobnie jak my robiliśmy w poprzednim odcinku).



Aby to wszystko mogło się wykonać, programiści musieli włożyć bardzo dużo wysiłku w stworzenie systemu (trudno powiedzieć programu) będącego w stanie przyjąć na siebie kontrolę nad komputerem. Jedna z pierwszych wersji tego systemu (KERNAL OS) została zainstalowana w komputerach Commodore PET. Ponieważ od początku było wiadomo, że system ten będzie musiał ulegać modyfikacjom zarówno ze względów rozwojowych jak i z powodu błędów (nikt nie jest doskonały) w nim zawartych, to też postanowiono, że zostanie zarezerwowany pewien (niezmienny) obszar w pamięci w którym umieszczona będzie tabela skoków. Po prostu jeżeli chcemy wywołać gotową procedurę np. drukującą znak na ekranie to nie szukamy pod jakim adresem w pamięci ona się znajduje, lecz odszukujemy jej pozycję w tabeli skoków i wywołujemy ją poprzez skok do tej tabeli. W przypadku zmiany systemu może się zdarzyć, że wywoływana przez nas procedura będzie leżała w innym obszarze pamięci. Mimo tego wywołanie poprzez tabelę spowoduje, że nasz program będzie dalej działał. Oryginalna tabela skoków (ta najwcześniejsza) zawierała 15 pozycji pod adresami $FFC0-$FFEA. W wersji dla C-64 została rozszerzona

do 39 pozycji ($FF81-$FFF5). Ciekawostką jest fakt, że korzystając tylko z oryginalnych 15 pozycji można np. napisać program, który może działać nie tylko na C-64, ale i na wcześniejszych modelach (PET, VC20). Uff! Tyle teorii... Do czego to się przyda? Wprawdzie nie da się napisać dobrej gry czy dema korzystając tylko z procedur systemu operacyjnego ale na pewno wiele razy nie będziemy musieli "wyważać otwartych drzwi" jeżeli skorzystamy z tego, że ktoś już kiedyś potrzebną nam akurat procedurę napisał i dał gotową w ROM'ie. Zacznijmy znowu od napisu na ekranie.

```
.A5000 LDA #$00
.A5002 STA $D020
.A5005 STA $D021
```

To powinno stać się regułą na początku każdego programu (jeżeli nie chcemy męczyć oczu niebieskimi kolorami tła)

```
.A5008 LDA #$93
.A500A JSR $FFD2
```

Pod adresem $FFD2 znajduje się w tabeli skoków procedura o nazwie CHROUT. Procedura ta wysyła na aktualne urządzenie wyjściowe bajt o tej samej wartości co znajdujący się w akumulatorze w momencie jej wywołania. Normalnie, po włączeniu (lub resecie) aktualnym urządzeniem wyjściowym jest ekran, ale z tej samej procedury można skorzystać zapisując dane na taśmie, dyskietce lub np. wysyłając je poprzez modem do linii telefonicznej lub radiolinii (Packet Radio) czy po prostu drukując na drukarce. W naszym przypadku dopiero co włączyliśmy komputer, więc na pewno urządzeniem wyjściowym jest ekran. Ale co to za dziwna wartość ($93) którą na niego wysyłamy? Z poprzedniego odcinka naszego kursu wiemy już, że te dwie ostatnie linie spowodują nam "wyczyszczenie" ekranu. Ale dlaczego? Weźmy do ręki instrukcję od komputera. Na końcu niej znajdują się tzw.dodatki (Appendix, Anhang). Jednym z nich jest tabela kodów ASCII (a dokładniej PETASCII) Odszukajmy w niej tą dziwaczną wartość ($93 to dziesiętnie 147). Zależnie od wersji językowej instrukcji, będziemy mieli przy wartości 147 napisane różne słowa lub

Tabela skoków systemu operacyjnego KERNAL dla C-64		
Adres	*Nazwa*	*Opis*
$FF81	CINT	inicjalizacja edytora ekranowego i procesora wizyjnego
$FF84	IOINIT	inicjalizacja urządzeń I/O (we/wy)
$FF87	RAMTAS	inicjalizacja pamięci RAM, bufora magnetofonu oraz ekranu
$FF8A	RESTOR	przywrócenie oryginalnych wartości wektorów I/O
$FF8D	VECTOR	odczytanie/ustalenie tabeli wektorów I/O
$FF90	SETMSG	ustalenie statusu informacji KERNAL'a
$FF93	SECOND	wysłanie adresu wtórnego po LISTEN
$FF96	TKSA	wysłanie adresu wtórnego po TALK
$FF99	MEMTOP	odczytanie/ustawienie wskaźnika końca pamięci
$FF9C	MEMBOT	odczytanie/ustawienie wskaźnika początku pamięci
$FF9F	SCNKEY	odczytanie klawiatury
$FFA2	SETTMO	ustawienie wskaźnika TIMEOUT dla łącza IEEE
$FFA5	ACPTR	przyjęcie bajtu z łącza SERIAL
$FFA8	CIOUT	wysłanie bajtu na łącze SERIAL
$FFAB	UNTLK	rozkaz dla urządzenia serial - UNTALK
$FFAE	UNLSN	rozkaz dla urządzenia serial - UNLISTEN
$FFB1	LISTEN	rozkaz dla urządzenia serial - LISTEN
$FFB4	TALK	rozkaz dla urządzenia serial - TALK
$FFB7	READST	odczytanie słowa statusu I/O
$FFBA	SETLFS	ustawienie parametrów kanału logicznego
$FFBD	SETNAM	ustawienie parametrów nazwy zbioru
$FFC0	OPEN	otworzenie kanału logicznego
$FFC3	CLOSE	zamknięcie kanału logicznego
$FFC6	CHKIN	zdefiniowanie kanału wejścia
$FFC9	CHKOUT	zdefiniowanie kanału wyjścia
$FFCC	CLRCHN	przywrócenie oryginalnych urządzeń we/wy
$FFCF	CHRIN	przyjęcie znaku z aktualnego kanału wejścia
$FFD2	CHROUT	wysłanie znaku do aktualnego kanału wyjścia
$FFD5	LOAD	ładowanie z urządzenia zewnętrznego
$FFD8	SAVE	zgrywanie na urządzenie zewnętrzne
$FFDB	SETTIM	ustawienie zegara programowego
$FFDE	RDTIM	odczytanie zegara programowego
$FFE1	STOP	sprawdzenie klawisza STOP
$FFE4	GETIN	przyjęcie znaku
$FFE7	CLALL	zamknięcie wszystkich zbiorów i kanałów
$FFEA	UDTIM	uaktualnienie zegara programowego
$FFED	SCREEN	odczytanie ilości wierszy i kolumn na ekranie
$FFF0	PLOT	odczytanie/ustalenie pozycji kursora
$FFE3	IOBASE	odczytanie adresu wyjściowego urządzeń I/O

Tabela skoków, do której nieraz jeszcze będziemy się odwoływać

znaczki ale niezależnie od języka znaczą one to samo: Czyścić ekran. Zauważyliśmy też pewnie zbieżność wartości 147 z BASIC'owym PRINT CHR$(147), który wykonuje dokładnie to samo. Tak! To ta sama procedura jest wywoływana w efekcie końcowym wykonania rozkazu PRINT. Dobrze! Mamy już wyczyszczony ekran. Teraz chcielibyśmy uzyskać napis najlepiej w kolorze białym. Zajrzyjmy znowu do tabeli ASCII i odszukajmy (na samym początku) kod odpowiadający za zmianę koloru kursora na biały. Kod ten to liczba pięć ($05). No to piszemy:

```
.A500D LDA #$05
.A500F JSR $FFD2
```

Teraz weźmy się za właściwy napis. Możemy to zrobić np. tak:

```
.A5012 LDA #$4B
.A5014 JSR $FFD2
.A5017 LDA #$45
.A5019 JSR $FFD2
.A501C LDA #$42
.A501E JSR $FFD2
.A5020 LDA #$41
.A5022 JSR $FFD2
.A5025 LDA #$42
.A5027 JSR $FFD2
.A502A RTS
```

Ale znacznie lepiej jest zrobić to w odpowiedniej "pętli". W poprzednim odcinku już takie rzeczy robiliśmy. Zatem do dzieła! Napiszmy od nowa:

```
.A5000 LDA #$00
.A5002 STA $D020
.A5005 STA $D021
```
wiadomo po co!
```
.A5008 TAX
.A5009 LDA $5100,X
.A500C BEQ $5015
.A500E JSR $FFD2
.A5011 INX
.A5012 JMP $5009
.A5015 RTS
```

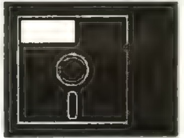
No! Od razu lepiej! TAX - wiemy co to jest (patrz odc. 3). Wiemy też, że w akumulatorze mieliśmy $00 więc i w X będziemy mieli $00. LDA $????,X - tego jeszcze nie było ale było już STA $????,X Jest to ten sam tryb adresowania (tzw. adresowanie absolutne z indeksem X) tyle tylko, że zamiast umieszczać kopię zawartości akumulatora pod adresem składającym się z sumy wartości $???? izawartości rejestru X (STA), umieszczamy kopię zawartości tego adresu w akumulatorze. W naszym przypadku adresem będzie $5100 + wartość z rejestru X. Następna linia: jeżeli wynik ostatniej operacji jest równy (zero) - skocz do adresu $5015. Pod aresem $5015 mamy RTS, czyli koniec naszego programu. Oznacza to, że jeżeli wykonując LDA $5100,X pobraliśmy do akumulatora wartość $00 to kończymy program. W innym przypadku wykonujemy JSR $FFD2 czyli wywołujemy z tabeli skoków procedurę CHROUT mając w akumulatorze gotową do przesłania wartość. INX powoduje zwiększenie rejestru X o jeden a JMP to po prostu skrót od (J)u(MP) - po polsku: skocz! Pod adres $5009 oczywiście...! Tam pobierzemy do akumulatora kolejny kod i jeżeli nie będzie to $00 to znowu wyślemy ją na ekran... itd. W ten sposób możemy wydrukować na ekranie do 256 znaków. To już dosyć sporo. Ale, Ale...! zapomnieliśmy jeszcze o jednym. Co będziemy pobierać spod adresu $5100 jeżeli nie umieściliśmy tam tego co chcemy zobaczyć na ekranie? Musimy jeszcze wpisać:

```
.M5100 93 05 4B 45 42 41
42 00
```

Widać, że w ten sposób umieściliśmy od adresu $5100 w górę nie tylko kody oznaczające sam napis ($4B, $45, $42, $41, $42) ale również te, które odpowiadały za czyszczenie ekranu ($93), zmianę koloru kursora ($05) i na końcu kod sterujący dla naszego programu informujący go o końcu tekstu ($00). Polecam poeksperymentowanie z różnymi kodami (i tekstami) umieszczanymi od adresu $5100. Pamiętać przy tym należy, że nie powinno ich być więcej niż 256 razem z wartością $00 umieszczoną na końcu tekstu.

Do zapamiętania!

LDA $????,X - ładowanie akumulatora w trybie "absolute with index X"
JMP $???? - skok bezwarunkowy pod adres absolutny (absolute address)
JSR $FFD2 - wywołanie procedury CHROUT (jednej z najczęściej używanych)



Słowniczek

KERNAL - tak nazywa się system operacyjny Commodore 64

CHARACTER GENERATOR - generator znaków tu znajdują się bajty określające wygląd każdego znaku (również liter i cyfr)

BASIC - (B)eginners (A)ll purpose (S)ymbolic (I)nstruction (C)ode - język programowania, który powinniśmy znać zanim zabierzemy się do nauki assemblera

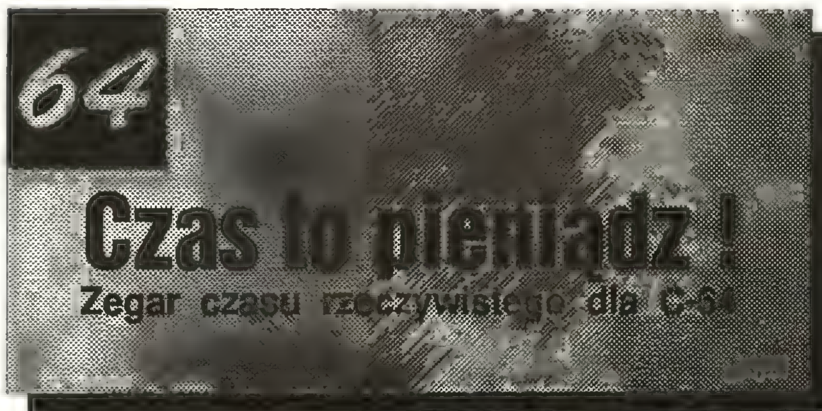
COMMODORE PET - Jeden z pierwszych komputerów domowych (osobistych) który pojawił się na rynku w roku 1980. Słowem "pet" określa się w języku angielskim np. ulubione zwierzątko domowe.

PETASCII - Zestaw kodów opierający się na tabeli ASCII jednakże z pewnymi istotnymi różnicami najważniejszą z nich było zamienienie miejscami kodów odpowiadających za duże i małe litery. Wszystkie ośmiobitowe modele komputerów Commodore (z C-64 włącznie) opierają się właśnie na tym standardzie.

ASCII - (A)merican (S)tandard (C)ode for (I)nformation (I)nterchange. Kod w którym każdemu znakowi (literowemu bądź funkcyjnemu) została przyporządkowana odpowiednia liczba. Przyjęcie tego standardu przez większość producentów umożliwia w miarę swobodną wymianę informacji pomiędzy różnymi systemami



Prom kosmiczny podczas startu...



Przeprowadziłem ostatnio kilka rozmów z Czytelnikami, którzy byli zainteresowani realizacją na Commodore 64 funkcji zegara czasu rzeczywistego (przy okazji chciałbym im serdecznie za te rozmowy podziękować, gdyż były one powodem powstania tego artykułu).

Pytania były różne, jedne dotyczyły sposobów wykorzystania zmiennej TI$ z poziomu BASIC, inne nawet zasady obsługi zegara czasu rzeczywistego, który obsługiwany jest przez układ CIA#1 (ang. Time Of Day, czyli Czas Dnia, dalej posługiwać się będziemy skrótem TOD).

Z poziomu języka BASIC sprawa jest względnie prosta: do dyspozycji mamy gotową zmienną systemową o nazwie TI$. Jak widać, jest to zmienna tekstowa (znak $ na końcu nazwy), w której aktualizowany jest przez system zegar czasu rzeczywistego. O ile nie ustawiono go inaczej, w TI$ możemy odczytać czas, jaki upłynął od momentu włączenia komputera.

Jak jednak korzystać z tej zmiennej? Bardzo prosto - odczyt wskazania zegara jest po prostu tekstem i można z nim robić np. takie operacje:

```
PRINT TI$
A$=TI$
H$=LEFT$(TI$,2):M$=MID$(TI$,2,2):
S$=RIGHT$(TI$,2)
```

Po bezpośrednim odczycie zmiennej (np. przez PRINT TI$), dane o czasie otrzymujemy w takiej oto przykładowej postaci: "013412". Jak to należy odczytać? Prosto: dwie pierwsze cyfry oznaczają liczbę godzin (tu: 01), dwie środkowe liczbę minut (34) a dwie ostatnie - liczbę sekund.

Fajnie, ale jak wpisać aktualną godzinę? W BASIC jest to również proste: przyjmijmy, że mamy teraz godzinę 21:22. Aby wpisać tą godzinę do TI$, należy zrobić z tego zapis właściwy dla tej zmiennej, czyli "212200". Następnie wpisując ten łańcuch tekstowy do TI$ (np. TI$="212200") powodujemy ustawienie zegara na "naszą" godzinę i jego start.

Ta zmienna sprawdza się zadowalająco w przypadku programów w BASIC, i w zasadzie jedynym mankamentem jest fakt, że chcąc mieć na ekranie stały odczyt czasu, należy co chwilę drukować zawartość TI$ na ekranie. Może to się jednak mocno znudzić, gdy będąc w trybie edycji będziemy pisać co chwila "PRINT TI$", od czego mogą nas porządnie rozboleć palce.

Tyle o zegarze TI$. Przejdźmy teraz do zegara czasu rzeczywistego TOD, o którym wspomniałem na początku artykułu. Rejestry obsługi tego zegara znajdują się w układzie CIA#1. Jest on w stanie odliczać czas z dokładnością do 1/10 sekundy. A oto opis rejestrów i ich adresy:

```
$DC08 (#56328) - 1/10 sekundy
$DC09 (#56329) - sekundy
$DC0A (#56330) - minuty
$DC0B (#56331) - godziny i wskaznik AM/PM.
```

Zasada jest prosta: odczytując te rejestry pobieramy dokładne wskazanie czasu; zapisując - ustawiamy zegar. Zapisu i odczytu dokonujemy posługując się systemem BCD, który pozwala na względnie łatwe operowanie danymi. Na czym on polega w przypadku TOD? Jak zauważyliśmy, rejestry zegara są jednobajtowe, tzn. że np. liczbę minut należy zapisać w jednym bajcie. Aha, powiecie, to proste - jeżeli chcę zapisać np. 21 minut, to do tego rejestru należy wprowadzić szesnastkową wartość $15 (dziesiętnie - 21). Niestety nie. Rejestry te przypominają pracą procesor w dziesiętnym trybie pracy, tzn. zapisując wspomniane wyżej 21 minut, należy wpisać do rejestru $DC0A szesnastkową wartość $21, czyli w ten sposób, jakby zapisywać liczby szesnastkowe tak, jak "widać" liczby dziesiętne. Hmm... myślę, że ktoś to zrozumiał. Dla ułatwienia, program w BASIC:

```
10 a$=''21":rem liczba dziesietna
20 w1=asc(left$(a$,1))-48
30 w2=asc(right$(a$,1))-48
40 a=w1*16+w2:rem wynik w BCD jest w `a'
```

Z zapisem i odczytem TOD są związane pewne zasady: zapisywać należy począwszy od rejestru godzin, poprzez minuty sekundy i kończyć na 1/10 sekundy. Zapis do rejestru godzin powoduje zatrzymanie zegara a zapis do rejestru 1/10 sekund powoduje wznowienie jego pracy z nowymi danymi. Podczas odczytu kolejność jest taka sama. Dla uniknięcia błędów, układ CIA#1 w momencie odczytu godzin "zamraża" stan zegara, tzn. stan rejestrów TOD'a pozostaje z takimi wartościami, z jakimi zastała go operacja odczytania rejestru $DC0B. Nie należy jednak utożsamiać tego z zatrzymaniem zegara - on pracuje nadal, z tym tylko, że dane w rejestrach nie są uaktualniane. Funkcję tą nazywamy "latching". Kontynuacja aktualizacji danych następuje z momentem odczytania rejestru 1/10 sekundy. Dlaczego jest to takie ważne? Wyobraźmy sobie, że o ile zegar nie zostałby "zatrzymany", to możliwe byłoby powstanie błedu odczytu, polegającego np. na tym, że w momencie odczytu godzin TOD mógłby wskazywać godzinę 9:59:59. Pomiędzy odczytem godzin a odczytem minut zegar mógłby zdążyć przestawić się na godzinę 10:00:00 i w tym momencie odczytane przez nasz program wskazanie dałoby w efekcie godz. 9:00:00, co raczej poprawne by nie było.

Jeszcze jedna mała uwaga: TOD jest zegarem 24-godzinnym, ale zapis

danych przeprowadza się w systemie używanym np. w Wielkiej Brytanii, tzn. zamiast godziny 23:00 piszemy 11:00 PM (po południu). Wkaźnik AM/PM jest najstarszym bitem w rejestrze godzin (czyli bit o wartości $80).

Dla zilustrowania zasady działania TOD'a pozwoliłem sobie napisać mały programik realizujący wyświetlanie zegarka cyfrowego w prawym górnym rogu ekranu. Podczas pisania tej procedury zrobiłem kilka założeń:

- wyświetlanie zegara powinno być zrealizowane bez użycia sprite'ów (powinny zostać do dyspozycji użytkownika).

- zegar powinien pracować korzystając z systemu przerwań komputera. Takie rozwiązanie pozwala na stworzenie programu, który nie będzie zanadto przeszkadzał systemowi w normalnej pracy.

- wyświetlanie danych na ekranie nie powinno przeszkadzać w edycji programów w BASIC itp.

W celu sprostania ostatniemu założeniu, wskazania zegara wyświetlane są na ekranie tylko w tym czasie, w którym ów fragment obrazu jest aktualnie generowany pod postacią sygnału wizyjnego. W momencie, w którym wiązka elektronów "zabiera się" za wyświetlanie drugiej linii znakowej ekranu, w miejsce wyświetlanego zegara wpisywane są uprzednio zapamiętane dane tak, aby system niczego nie "zauważył".

Program został wydrukowany w postaci dwóch listingów: jednego w postaci listingu BASIC oraz tzw. postaci źródłowej spod TurboAssembler'a wraz z komentarzami.

Paweł Sołtysiński

64



Ten artykuł poświęcony jest miłośnikom cartridge'a Action Replay. Jest on powszechnie ceniony za jego prawie perfekcyjne możliwości "zamrażania" programów. Po wciśnięciu przycisku podpisanego "freezer" komputer automatycznie zatrzymuje wykonywanie programu i otwiera specjalne menu, z którego wybrać możemy interesujące nas opcje a potem powrócić do wykonywania zamrożonego programu. Poprzez wybranie odpowiedniej opcji można np. nagrać na dysk aktualnie znajdujący się na ekranie obrazek w trybie multicolor lub przejrzeć znajdujące się w pamięci sprite'y. Mi osobiście bardzo brakowało jeszcze funkcji wyszukiwania generatorów znaków (danych o kształtach znaków) i wyszukiwania znajdujących się w pamięci sampli. Cała sztuka polegała by jednak na tym, by po wyszukaniu owych interesujących nas danych, można było kontynuować pracę programu. Nowe funkcje oznaczają dodatkowe programy, które będą je realizować, a które, niestety, będzie należało wgrać do pamięci, co powinno zmienić jej zawartość i uniemożliwić kontynuację zatrzymanego programu. Jak to więc zrobić?

Jak się jednak okazało, zadanie nie było takie trudne. Proponuję mały eksperyment:

a) wgrajcie do pamięci dowolny program i uruchomcie go;

b) zatrzymajcie jego pracę przyciskiem "freeze";

c) w czasie gdy będziecie w menu "zamrażacza" - przyśnijcie RESET i wybierzcie opcję NORMAL RESET (F-3);

d) na ekranie pojawił się znajomy napis "Commodore Basic v2...." - możemy się teraz trochę pobawić w BASIC (nie za dużo!);

e) wciśnijcie ponownie RESET i zamiast wybierać którąś z możliwości przypisanych klawiszom funkcyjnym - wciśnijcie klawisz RESTORE (w starych wersjach Commodore 64 należy to uczynić z pewną werwą...);

f) no i znaleźliśmy się z powrotem w... "freeze menu"! Można teraz wybrać opcję RESTART i nasz wgrany uprzednio program będzie kontynuowany.

Jak to możliwe? Jak się okazuje, w momencie wciśnięcia przycisku FREEZE, cartridge przepisuje pamięć od adresu 0 do adresu $0A00 do własnej pamięci. Podczas wykonywania opcji RESTART jest ona ponownie przepisywana do RAM i podjęte zostaje wykonywanie programu. Dlatego gdy wybraliśmy "NORMAL RESET", zmiany w pamięci dotyczyły tylko obszaru od 0 do $0803, co przy restacie z "zamrażacza" zostało naprawione przez skopiowanie dawnych wartości z pamięci własnej Action Replay.



Wykorzystując ten fakt, napisałem dwa programy, które wgrywane z dysku automatycznie wgrywają się w "bezpieczny" obszar ekranu i uruchamiają. Pierwszy z nich to "1AR-TOOL", którego zadaniem jest odnalezienie i ewentualne nagranie na dysk generatorów znaków, a drugi to "2AR-TOOL", który wycina nam z pamięci sample (digitalizowane dźwięki). Znajdujące się na końcu KEBABA listingi są tzw. programami instalacyjnymi, tzn. po ich uruchomieniu zapisywane są na dysku właściwe programy. Jest to konieczne ze względu na specjalny rodzaj samouruchomienia, w które programy te zostały wyposażone. Oba listingi wpisać można do pamięci posługując się Korektorem lub dowolnym monitorem (pomijamy wtedy sumy kontrolne, które wydrukowane są w nawiasach).



A teraz zasady użycia:

a) wgrać interesujący nas program i uruchomić go;

b) wcisnąć FREEZE;

c) wcisnąć RESET i wybrać opcję NORMAL RESET;

d) wpisać w trybie bezpośrednim BASIC instrukcję POKE 52,10 (wyjaśnienie dalej);

e) załadować jeden z progrmów,

np. LOAD"1AR-TOOL",8,1

f) po samouruchomieniu się programu wyszukać przy jego pomocy interesujące nas dane (instrukcje na ekranie);

g) po skończeniu pracy wcisnąć RESET i uderzy w klawisz RESTORE;

i) przywrócić program do pracy przez wybranie opcji RESTART.

A wracając do owego POKE 52,10 - podana w komendzie LOAD nazwa ładowanego programu jest przez komputer traktowana jako zmienna tekstowa i w związku z tym jest tymczasowo zapisywana w pamięci od adresu $9FFF w dół, a więc w obszarze "nie chronionym". Modyfikując zmienną systemową powodujemy, że nazwa będzie umieszczona od adresu $09FF w dół, co nie uszkodzi zawartości komórek o wyższych adresach.

Paweł Sołtysiński

Ogłoszenia drobne

Sprzedam Amiga 500, modulator, joystick, dyski, cena 9,9 mln.
Piotr Nowak,
ul. Prądzyńskiego 47/27 Poznań

Zamienię Final II na Black Box 8.
Maciej Wajman
Pszczew, tel. 280

Sprzedam Black Box v.8, 135 tys. zł.
Maciej Górtowski,
al. Żołnierza Boczna 8/4, Stargard

Zainteresowanym odstąpię prywatny zbiór 1000 dyskietek na komputer C-64. M.G., ul. Dekutowskiego 14/23, 39-400 Tarnobrzeg

Sprzedam C-64II, magnetofon, monitor (zielony), drukarkę, Final 3, expander, 1000 programów.
Robert Sadkowski,
ul. Bema 37/16 Nowogard

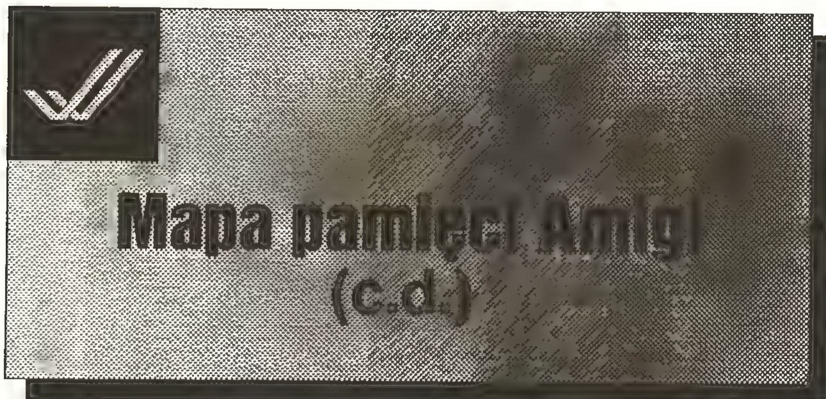
Sprzedam tanio monitor Neptun 156b.
Zbigniew Kułakowski,
Zamość, ul. Słowicza 4, tel. 54-52

Wymienię literaturę, osprzęt, oprogramowanie (C-64, Amiga).
Józef Młynarczyk,
21-044 Trawniki (Dom Nauczyciela)



Od około czterech miesięcy, amigowcy na całym świecie mogą cieszyć się nowym pakietem do programowania w Pascal'u. HighSpeed Pascal 1.00, wydany przez firmę HiSoft, na pewno zdobędzie uznanie wśród jego przyszłych użytkowników, tym bardziej, że poprzednie kompilatory tego popularnego, szczególnie w "pecetowskich" kręgach, języka nie nadawały się praktycznie do pracy. Wolny kompilator KickPascal'a i skandalicznie wykonany edytor dyskwalifikowały go już na samym starcie. Obecnie wspomniane niedoogodności zostały w zupełności wyeliminowane. HighSpeed Pascal 1.00 cechuje się znakomitym edytorem, wykonanym oczywiście według przyjętego standardu tzw. "New Look", pozwalającym na pracę nad kilkoma projektami jednocześnie w niezależnych od siebie oknach. Takie rozwiązanie, niemal niezbędne przy pracy nad dużymi programami, zapewnia programiście komfort i wygodę pracy, tym bardziej, że okienka mogą być automatycznie sortowane i przemieszczane na różne sposoby (pionowo, horyzontalnie, po przekątnej, itp.). Poza standardowymi operacjami na blokach tekstu (Copy, Cut, Paste), oraz opcjami przeszukiwania (Search) i wymiany (Replace) ciągów znaków istnieje komenda Undo pozwalająca na odtworzenie nieopatrznie skasowanej linii programu. Na szczególną uwagę zasługuje możliwość praktycznie całkowitej kontroli nad programem, bez konieczności odrywania rąk od klawiatury i sięgania do myszy, albowiem na wszystkie pytania programu możemy odpowiadać poprzez naciśnięcie klawisza z literą podkreśloną w interesującej nas odpowiedzi. Edytor współpracujący z debuger'em i kompilatorem można w bardzo prosty sposób skonfigurować, według własnych potrzeb, dzięki bogatym preferencjom. Przejdźmy teraz do serca pakietu, czyli kompilatora. Tutaj wielka niespodzianka. Okazuje się, że jest on niemalże w pełni kompatybilny z Borland'owskim Turbo Pascal'em 5.0, a konkretnie z podstawowym standardem języka, oraz modułami System, Dos, Crt, Graph, odpowiedzialnymi kolejno za procedury ogólnego zastosowania, wejścia/wyjścia, pomostu między użytkownikiem a maszyną, grafiki. Piszę "niemalże", gdyż podczas testowania wypadły małe "perełki", jak na przykład drobne niezgodności trybów graficznych wynikające jednak głównie z różnic sprzętowych, czy nierozpoznanie procedury Sound z modułu Crt. Cóż, takie drobiazgi można chyba wybaczyć autorom pierwszej wersji programu, tym bardziej że emulacja efektów dźwiękowych "peceтów" jest rzeczywiście nie lada sztuką. Oprócz wyżej wspomnianych modułów istnieje możliwość wykorzystania wszystkich procedur systemu operacyjnego włączając w to zarówno biblioteki jak i urządzenia. Autorzy znając dokładnie potęgę amigowskiego ROM'u dołożyli starań, aby korzystanie z niego było jak najłatwiejsze, dlatego też załączyli całą gamę przykładowych procedur źródłowych.

Krzysztof Kobus



Na wstępie chciałbym wyjaśnić małe nieporozumienie. Otóż chochlik zamienił nam w poprzednim Kebabie odcinki cyklu w związku z czym wydrukowaliśmy niedokończonyy odcinek czwarty (zamiast trzeciego). Bardzo przepraszamy wszystkich Czytelników i dziś prezentujemy zaległy materiał.

00E CLXDAT O * D

Rejestr odczytu kolizji. Za pomocą tego rejestru możemy kontrolować zaistniałe kolizje czyli zderzenia między poszczególnymi objektami. Mogą zaistnieć trzy sytuacje. Pierwsza, gdy zderzą się ze sobą dwa Sprite'y (pocisk uderzający w ruchomy objekt), druga gdy Sprite uderza o określone Bit-Plane'y (statek kosmiczny jadący w tunelu), oraz trzecia, gdy objekty zdefiniowane na Bit-plane'ach animowane Blitterem kolidują ze sobą (najczęściej duże, wielokolorowe "potwory" w grach). Procesor graficzny przygotowując obraz do wyświetlania na podstawie danych zawartych w pamięci, sprawdza czy określone obiekty nie nachodzą na siebie. Jeśli tak się stanie ustawia odpowiedzialny bit za dane zderzenie w tymże rejestrze. I tak poszczególne bity oznaczają:

bit 15 Nie używany.
bit 14 Sprite 4 (lub 5) koliduje ze Sprite'm 6 (lub 7).
bit 13 Sprite 2 (lub 3) koliduje ze Sprite'm 6 (lub 7).
bit 12 Sprite 2 (lub 3) koliduje ze Sprite'm 4 (lub 5).
bit 11 Sprite 0 (lub 1) koliduje ze Sprite'm 6 (lub 7).
bit 10 Sprite 0 (lub 1) koliduje ze Sprite'm 4 (lub 5).
bit 9 Sprite 0 (lub 1) koliduje ze Sprite'm 2 (lub 3).
bit 8 Pole 0 koliduje ze Sprite'm 6 (lub 7).
bit 7 Pole 0 koliduje ze Sprite'm 4 (lub 5).
bit 6 Pole 0 koliduje ze Sprite'm 2 (lub 3).
bit 5 Pole 0 koliduje ze Sprite'm 0 (lub 1).
bit 4 Pole 1 koliduje ze Sprite'm 6 (lub 7).
bit 3 Pole 1 koliduje ze Sprite'm 4 (lub 5).
bit 2 Pole 1 koliduje ze Sprite'm 2 (lub 3).
bit 1 Pole 1 koliduje ze Sprite'm 0 (lub 1).
bit 0 Pole 1 koliduje z Polem 0.

Pole 0 oznacza Bit-Plane'y o parzystych numerach (2,4,6), natomiast Pole 1 to Bit-Plane'y oznaczone liczbami nieparzystymi (1,3,5). Rejestr ten zostaje automatycznie wyzerowany zaraz, gdy zostanie odczytany. Zderzenia z Polami będą rejestrowane bez względu na to czy tryb Dual-Playfield będzie aktywny czy wyłączony.

010 ADKCONR O * P

Kontroler dysku i audio - odczyt. Rejestr ten odpowiedzialny jest za pracę dysku oraz kanałów audio. Ustawienie odpowiednich bitów znacznie poszerza możliwości muzyczne Amigi. Pozwala na używanie jednego z kanałów jako modulatora drugiego, przy czym modulacja może się odbywać zarówno w zakresie amplitudy jak i częstotliwości. Zmiany amplitudy fali dźwiękowej prowadzą do zmian jej głośności, natomiast modulacja częstotliwości powoduje zwiększenie lub zmniejszenie jej okresu. Poszczególne bity oznaczają:

bit 15 (SET/CLR) Wskazuje czy pozostałe bity rejestru oznaczone logiczną jedynką, zostaną zinterpretowane przez wewnętrzne układy koputera jako ustawione lub skasowane. Zostanie dokładnie opisany przy ADKCON.
bit 14 (PRECOMP1) Bardziej znaczący bit transmisji (MSB).
bit 13 (PRECOMP0) Mniej znaczący bit transmisji (LSB). Odpowiednia kombinacja tych bitów daje różny czas opóźnienia dostępu do danych: 00 - brak opóźnienia, 01 - 140ns, 10 - 280ns, 11 - 560ns.



bit 12 (MFMPREC) Wybór trybu transmisji. Ustawienie przełącza na tryb MFM, natomiast 0 na GCR.
bit 11 (UARTBRK) Używane przy transmisji przez port szeregowy.
bit 10 (WORDSYNC) Ustawiony powoduje synchronizację i rozpoczęcie przez DMA odczytu od danego słowa. Słowo to musi uprzednio zostać zapisane w komórce DSKSYNC.
bit 9 (MSBSYNC) Używane w trybie GCR. Ustawienie synchronizacji względem MSB.



bit 8 (FAST) Ustawienie oznacza czas 2 mikrosekund na odczytanie sekwencji bitów (używane w trybie MFM), natomiast wyzerowanie przełącza na czas 4 mikrosekund (używane w trybie GCR).
bit 7 (ATPER3) Wyłącza kanał 3.
bit 6 (ATPER2) Przydzielenie kanału 2 dla modulacji częstotliwości kanału 3.
bit 5 (ATPER1) Przydzielenie kanału 1 dla modulacji częstotliwości kanału 2.
bit 4 (ATPER0) Przydzielenie kanału 0 dla modulacji częstotliwości kanału 1.
bit 3 (ATVOL3) Wyłącza kanał 3.
bit 2 (ATVOL2) Przydzielenie kanału 2 dla modulacji amplitudy kanału 3.
bit 1 (ATVOL1) Przydzielenie kanału 1 dla modulacji amplitudy kanału 2.
bit 0 (ATVOL0) Przydzielenie kanału 0 dla modulacji amplitudy kanału 1.

Twórcy Amigi, wprowadzając możliwość pracy dysku w trybach MFM iGCR maksymalnie uprościli problem wymiany informacji dyskowych, zapisanych w formacie Macintosh'a, C64 i kompatybilnych z IBM. Dokładny opis kodowania w tych trybach wykracza poza ramy "Mapy pamięci" i być może zostanie szerzej opisany w przyszłości na łamach Kebaba.

012 POT0DAT O * P
014 POT1DAT O * P

Liczniki manipulatorów analogowych (ang: paddle) numer 0 i 1. Rejestry te pozwalają na odczyt proporcjonalnych wartości manipulatorów podłączonych do portów 0 i 1. Poszczególne bity oznaczają:

bity 15-8 Przesunięcie pionowe - Y
bity 7-0 Przesunięcie poziome - X

016 POTGOR O * P

Odczyt kontrolera dwukierunkowego, czterobitowego portu wejścia/ wyjścia. Rejesr ten kontroluje komunikację Amigi przez porty wejścia/ wyjścia. Poszczególne bity mają następujące znaczenie:
bit 15 (OUTRY) Ustawienie oznacza umożliwienie komunikacji przez bit 14 tegoż rejestru.
bit 14 (DATRY) Bit danych.
bit 13 (OUTRX) Ustawienie oznacza umożliwienie komunikacji przez bit 12 tegoż rejestru.
bit 12 (DATRX) Bit danych.
bit 11 (OUTLY) Ustawienie oznacza umożliwienie komunikacji przez bit 10 tegoż rejestru.
bit 10 (DATLY) Bit danych. Normalnie oznacza prawy przycisk myszy.
bit 9 (OUTLY) Ustawienie oznacza umożliwienie komunikacji przez bit 8 tegoż rejestru.
bit 8 (DATLY) Bit danych. W przypadku podłączenia myszy z trzema przyciskami odpowiada za stan środkowego.
bity 7-1 (X) Numer identyfikacyjny układu.
bit 0 (START) Resetuje i startuje liczniki POT0DAT i POT1DAT.

Jak już wspomniałem bit 10 odpowiada za stan prawego przycisku myszy. Logiczna jedynka świadczy o tym, że jest on zwolniony, natomiast zero, że jest przyciśnięty. Jego stan można przetestować w następujący sposób:

```
Loop: btst   #$0a,$dff016
      bne.s  Loop
      rts
```

Pętla zostanie zakończona w przypadku naciśnięcia prawego przycisku myszy.

018 SERDATR O * P

Odczyt danych z portu szeregowego i jego status. Cóż takiego kryje się pod tajemniczą nazwą port szeregowy? Otóż nic innego jak interfejs służący do szeregowej wymiany informacji, (czyli bit za bitem w ściśle określonych odstępach czasowych, w odróżnieniu do transmisji równoległej, gdzie przesyłanie odbywa się całymi bajtami - poszczególne bity na innych liniach elektrycznych) między komputerem a urządzeniem zewnętrznym, na przykład modemem. Oczywiście przesyłanie może odbywać sie w obu kierunkach w ogromnym zakresie częstotliwości - od 110 do ponad 1000000 bitów na sekundę i porcjami po 8 lub 9 bitów. Amigowski port szeregowy, podobnie jak w większości komputerów pracuje w szeroko rozpowszechnionym standardzie RS-232. Gniazdo tego interfejsu znajduje się z tyłu komputera (Serial Port). Jak już wspomniałem rejestr ten służy głównie do odczytu danych transmitowanych przez "Serial". Wiadomo też, że bity "przychodzą" do niego pojedynczo, co jest bardzo niewygodne przy obróbce przez procesor. Dlatego dopiero po pojawieniu się odpowiedniej ilości bitów (8 lub 9) generowane jest przerwanie, mówiące o zapełnieniu rejestru pośredniego gromadzącego kolejne bity, tym samym informujące procesor o konieczności odczytu tegoż rejestru. Poszczególne bity oznaczają:
bit 15 (OVRUN) Jego ustawienie oznacza przepełnienie rejestru danych, czyli że przybył kolejny bit, zanim poprzednia porcja informacji została odczytana.
bit 14 (RBF) Ustawiony oznacza, że zgromadzony został komplet bitów przy odczycie i bity danych mogą zostać odczytane.
bit 13 (TBF) Ustawiony oznacza, że komplet bitów został odczytany i może być przyjęta kolejna porcja informacji.
bit 12 (TSRE) Ustawiony oznacza, że rejestr gromadzący bity jest pusty.
bit 11 (RXD) Zawiera kopię sygnału z końcówki Pauli o tej samej nazwie.
bit 10 Nie używany.
bit 9 (STP) Używany przy transmisji 9-cio bitowej - znacznik STOP.
bit 8 (STP-DB8) Dla transmisji 8-mio bitowej znacznik STOP, dla 9-cio bitowej bit danych.
bit 7-0 (DB7-DB0) Bity danych.

01A DSKBYTR O * P

Rejestr transmisji danych z/na dysk i status odczytu. Podczas operacji dyskowych dane zostają przekazywane do/z pamięci właśnie prez ten rejestr. Poszczególne bity oznaczają:
bit 15 (DSKBYT) Ustawiony oznacza, że przybył kolejny bajt danych podczas odczytu z dyskietki. Bit ten zostanie automatycznie skasowany po odczytaniu tegoż rejestru.
bit 14 (DMAON) Logiczna jedynka świadczy o tym, że DMA dla dysku jest włączone - transmisja może się odbywać.
bit 13 (DISKWRITE) Jedynka oznacza operację zapisu, zero odczytu.
bit 12 (WORDEQUAL) Ustawiony oznacza, że dane czytane z dysku zostały zsynchronizowane z rejestrem DSKSYNC.
bity 11-8 Nie używane.
bity 7-0 Zawierają bajt danych aktualnie transmitowany na/z dysku.

01C INTENAR O * P

Odczyt bitów zezwalających na przerwania. Za pomocą tego rejestru możemy sprawdzić które przerwania mogą być wykonywane, a których wykonanie jest zabronione. Poszczególne bity podłączone są następująco:
bit 15 (SET/CLR) Wskazuje czy bity oznaczone 1 będą interpretowane jako ustawione, czy też skasowane.
bit 14 (INTEN) Skasowanie tego bitu zabrania wykonania jakiegokolwiek przerwania.
bit 13 (EXTER) Odpowiedzialny za przerwania urządzeń zewnętrzynch.
bit 12 (DSKSYN) Wywołanie przerwania w czasie synchronizacji przesyłu danych z lub na dysk.
bit 11 (RBF) Zgłasza gotowość przyjęcia danych na port szeregowy.
bit 10 (AUD3) Generacja przerwania po zakończeniu pracy kanału 3 i tym samym informacja o możliwości przyjęcia następnych danych.
bit 9 (AUD2) Analogicznie jak AUD3, dotyczy kanału 2.
bit 8 (AUD1) Analogicznie jak AUD3, dotyczy kanału 1.
bit 7 (AUD0) Analogicznie jak AUD3, dotyczy kanału 0.
bit 6 (BLIT) Generacja przerwania po zakończeniu pracy przez Blitter'a. Tym samym sygnalizuje gotowość do wykonania dalszych zadań przez ten układ.
bit 5 (VERTB) Generacja przerwania wygaszania pionowego czyli w momencie, gdy wiązka elektronów przemieszczana jest z prawego dolnego do lewego górnego rogu ekranu. Podczas tego przerwania system wykonuje wiele niezbędnych dla prawidłowej pracy komputera czynności (np obsługa myszy), a po ich wykonaniu użytkownik może zlecić dodatkowe zadania.
bit 4 (COPER) Zleca wykonanie przerwania, gdy wyświetlanie ekranu dojdzie do zadanej pozycji.
bit 3 (PORTS) Odpowiedzialny za przerwania we/wy, oraz zegarów.

bit 2 (SOFT) Zarezerwowany dla prerwań zdefiniowanych programowo.
bit 1 (DSKBLK) Generuje przerwanie po zakończonej pracy dysku.
bit 0 (TBE) Zgłasza konieczność wysłania/pobrania danych przez port szregowy.

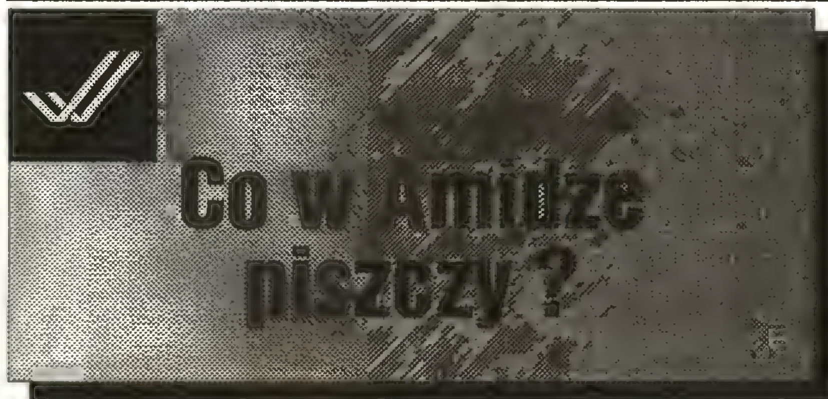
Bity ustawione oznaczają, iż przerwania mogą zostać wykonane.

01E INTREQR O * P

Rejestr ten zawiera informacje o aktualnie wykonywanych przerwaniach, bądź tych, które zaraz mają zostać wykonane. Bit ustawiony oznacza, że przerwanie jest zgłoszone. Znaczenie poszczególnych bitów jest takie same jak w rejestrze INTENAR.

W dziale "Listingi" znajdziecie procedurę inicjującą i obsługującą przerwania VBlank i Copper. Pierwsze z nich co 50 wywołań, czyli co sekundę miga diodą, natomiast drugie wywoływane jest dwa razy w ciągu ramki i powoduje chwilową zmianę koloru na niebieski. Przedstawione procedury wykonywane podczas danych przerwań są oczywiście trywialne w swoim działaniu, chodziło mi bowiem o pokazanie samego mechanizmu ich wywoływania, a Czytelnikowi pozostawiam napisanie konkretnych procedur ich wykorzystania. Pamiętajcie jednak, że zazwyczaj są one krytyczne czasowo i powinny realizować swoje zadania w jak najkrótszym czasie. Dodam jeszcze tylko, używane tam rejestry $09A i $09C mają analogiczne znaczenie jak $01C i 01E ale służą nie do odczytu, a zapisu.

Krzysztof Kobus



Dla nikogo pewnie nie jest tajemnicą, że każdy kolejny model Amigi różni się od swych poprzedniczek, czy to wyglądem płyty głównej, nowymi chip'ami czy nowszą wersją systemu operacyjnego. Wprowadzone zmiany powodują czasami komplikacje w pracy starszych programów, nie tolerujących modyfikacji. Kłopoty te występują głównie w programach, które wykorzystują w swej pracy określone cechy danego modelu komputera, korzystając np. z bezpośrednich skoków do ROM'u. Jeśli nie chcemy aby nasz program poszedł w ich ślady, to mamy do wyboru dwia kierunki postępowania: możemy albo zrezygnować z używania podobnych tricków (co przecież nie zawsze jest możliwe) lub też na samym początku naszego programu odczytać aktualną konfigurację systemu i w zależności od niej podjąć określone działania. Daje nam to możliwość nie tylko uniknięcia kłopotów w funkcjonowaniu programu lecz także pozwala na pełne wykorzystanie możliwości oferowanych przez dany komputer. Ktoś może powiedzieć: Dobrze, wszystko fajnie, tylko skąd ja mam wziąć informacje o konfiguracji?! Wbrew pozorom nie są to informacje ściśle tajne a wszystko co będzie wam potrzebne do ich zdobycia to Kebab z niniejszym artykułem oraz trochę doświadczenia w programowaniu.

Zacznijmy od najważniejszej rzeczy w komputerze, a mianowicie od jego "serca" czyli mikroprocesora. Określenie typu procesora na którym wykonywany jest nasz program jest na tyle ważne, że nowsze modele Motoroli dysponują poszerzoną, w stosunku do MC68000 listą rozkazów oraz większą mocą obliczeniową. Możemy więc zawczasu przygotować na przykład bibliotekę procedur dla danego procesora (np. z wykorzystaniem nowych rozkazów) i korzystać z niej w przypadku jego wykrycia. Interesującę nas informacje znajdziemy w komórce AttnFlags ($128 względem bazy "exec.library"). Testując jej odpowiednie bity oraz odpowiednio je interpretując otrzymamy informacje o typie procesora. Oto znaczenie poszczególnych bitów tej komórki:

bit 4 - koprocesor matematyczny 68881
bity 1 i 0 - MC68020 lub wyżej
Bit 0 - MC68010

Jeśli żaden bit nie jest ustawiony to nie oznacza to bynajmniej braku procesora (?!) a jedynie fakt istnienia poczciwego MC68000.

Kolejne modele Amig różnią się między sobą także poszczególnymi chip'ami pomocniczymi. Największe zmiany ewolucyjne dotyczą układu opowiedzialnego za grafikę i dzwięk. Zasadnicza różnica polega na zwiększeniu obszaru pamięci jaki może być obsługiwany przez ten układ (dostęp do Chip-Ram) oraz poszerzeniu listy rejestrów (ang: Custom Registers). Aby przekonać się z którą wersją mamy do czynienia wystarczy sprawdzić zawartość komórki VPOSR ($dff004). Konkretnie interesują nas bity 12 i 13. W rozszyfrowaniu ich znaczenia pomoże poniższa tabelka. ECS oznacza nową wersję danego układu (ang: Expanded Chip Set):

13	12	Agnus
0	0	PAL 8371
1	0	PAL ECS 8372
0	1	NTSC 8376
1	1	NTSC ECS 8372

Do określania w jakim systemie (NTSC czy PAL) pracuje aktualnie Amiga proponuję uciec się do następującej metody. W "exec.library" znajduje się komórka o nazwie VBlank-Frequency ($212 od bazy), której zawartość odpowiada częstotliwości wyświetlania obrazu. Jak już wiemy PAL pracuje na częstotliwości 50Hz zatem tą wartość znajdziemy w przypadku tego systemu. Natomiast w przypadku amerykańskiego systemu NTSC odczytana wartość równa będzie 60. Jest oczywiście jeszcze inna metoda i podejrzewam, że nie jest ostatnią. Wykorzystamy w tym celu "graphics.library", bowiem tam znajduje

się komórka (a właściwie dwa bajty) o nazwie NormalDisplayRows, która przechowuję liczbę wyświetlanych linii. W systemie NTSC maksymalna wartość tej komórki, a więc liczba linii, nie przekracza 200.

Do odczytu ilości wolnej pamięci wykorzystujemy systemową procedurę AvailMem z biblioteki "exec.library". Przed odwołaniem się do niej musimy w rejestrze d1 procesora umieścić znacznik typu pamięci jaki nas interesuje. Dozwolone są następujące wartości:

Wartość Nazwa Komentarz
$00001 PUBLIC Wolna pamięć całkowita.
$00002 CHIP Wolna pamięć typu CHIP.
$00004 FAST Wolna pamięć typu FAST.
$20000 LARGEST Największy obszar pamięci możliwy do zaalokowania.

Teraz należy tylko odwołać się do wspomnianej procedury, a w jej wyniku w rejestrze d0 otrzymamy ilość wolnej pamięci danego typu.

Przejdźmy do pamięci dyskowej. Wbudowany w Amigę kontroler umożliwia podłączenie i obsługę do 4 stacji dysków (wliczając także wbudowany napęd). Osobiście spotkałem się z kilkoma metodami programowego określania liczby aktualnie podłączonych stacji dysków, lecz w tym miejscu przedstawię najkrótszą z nich, wykorzystującą do tego celu strukturę DiskResource. Po otwarciu dostępu do niej (funkcją OpenResource z "exec.library") sprawdzamy zawartość kolejnych komórek UnitId (od adresu 48 względem początku struktury DiskResource) odpowiadającym poszczególnym napędom. Wszystkie 4 komórki mają długość czterech bajtów (długie słowo), zaś wartość zero w poszczególnej komórce należy interpretować jako "napęd podłączony".

Na zakończenie chciałbym zwrócić Waszą uwagę na program źródłowy zamieszczony na naszym majowym Kebab-Public-Domain Dysku. Podaje on podstawowe parametry konfiguracji systemu, bazując na podanych wyżej metodach, dodatkowo wykrywając KickStart 2.0. Zastosowana metoda, poprzez próbę otwarcia odpowiedniej, nie istniejącej w starym KickStarcie wersji biblioteki DOS jest może dość prymitywna, ale w tym przypadku w zupełności wystarczająca. Tu krótka uwaga dla osób chcących wykorzystać ten programik. Po wgraniu source'a oraz udanej asemblacji należy nagrać kod wynikowy na dysk (komendą "WO") i uruchomić jak każdy inny program, z okienka CLI. Użycie opcji "J" asemblera spowoduje zablokowanie systemu przy próbie wyprowadzenia konfiguracji na ekran.

Marcin Orłowski



W każdym odcinku naszego cyklu informacje z poprzedniego miesiąca będą rozbudowywane o nowe szczegóły. Ostatnim razem zacząłem omawiać podstawy używania zmiennych w języku BASIC V2. Dla przypomnienia, były to zmienne tekstowe (te ze znakiem "$" na końcu nazw) i zmienne liczbowe (te bez "$"). Istnieje jeszcze jeden rodzaj zmiennych - zmienne liczbowe całkowite. Do czego one służą? Stosuje się je zazwyczaj ponieważ podporządkowanie danej zmiennej określonej liczby powoduje zapamiętanie jej jako całkowitą bez konieczności dodatkowej obróbki, co czasami jest wygodniejsze.

Jak definiuje się takie zmienne? W prosty sposób - wystarczy po podaniu jej nazwy dodać na końcu znak "%", np. ABC%. Przykład wykorzystania:

```
10AB%=343.5733
20PRINT``WartoscAB%='';AB%
```

Po uruchomieniu otrzymamy na ekranie aktualną wartość zmiennej AB%, która wynosić będzie dokładnie 343 (jak widać, wartości po przecinku zostały zignorowane). I to by były w zasadzie wszystkie rodzaje zmiennych z wyjątkiem tzw. tablic, o korzystaniu z których napiszę następnym razem. Dziś zajmiemy się funkcjami, które służą do "obrabiania" zmiennych.

Jak wiadomo, istotą użycia komputerów jest ich zdolność do obróbki danych. Dotyczy to także zmiennych, których używamy przy programowaniu w języku BASIC. Oczywiście, możemy przetwarzać dane w nich zawarte poprzez własne procedury, w których wykorzystać możemy gotowe rozkazy j.BASIC, które służą do obsługi zmiennych. A oto większość z nich wraz z opisem i przykładami, które pomogą zrozumieć ich działanie:

DLA ZMIENNYCH LICZBOWYCH:

*) INT(argument) - wyznacza wartość całkwitą argumentu, np.:

```
10 A=35.89
20 B=INT(A):PRINT B

run
35
```

Należy pamiętać, że argument może być wyrażeniem, np. "2*18/C+8".

*) ABS(argument) - wyznacza wartość bezwzględną argumentu, np.:

```
10 A=45:B=-12
20 PRINT ABS(A), ABS(B)

run
45   12
```

Jak widać, w uproszczeniu funkcja ABS "ignoruje" ewentualny znak

minus - wynik jest zawsze dodatni.

*) STR$(argument) - transformuje zmienną liczbową w łańcuch tekstu, który potem może zostać wykorzystany do zdefiniowania zmiennej tekstowej, np.:

```
10 A=125
20 B$=STR$(A):PRINT B$

run
125
```

Funkcja STR$ tworzy łańcuch znaków odpowiadający kodom ASCII cyfr wykorzystanych do opisania liczby (argumentu). Należy zwrócić uwagę na fakt, iż na początku łańcucha " 125" znajduje się znak pusty (spacja), który w przypadku liczb ujemnych zastępowany jest przez znak "-" (minus).

DLA ZMIENNYCH TEKSTOWYCH:

*) LEN(tekst) - zwraca liczbę znaków tekstu (lub zmiennej tekstowej) umieszczonego w nawiasach, np.:

```
10A$=''abcde''
20 PRINT LEN(A$)
run
5
```

*) LEFT$(tekst,argument) - rozkaz, który pobiera określoną argumentem ilość znaków od lewej "strony" podanego tekstu (lub zmiennej), np.:

```
10A$=''abcde''
20 B$=LEFT$(A$,3)
30 PRINT B$

run
abc
```

Korzystając z funkcji LEFT$ mogliśmy stworzyć zmienną B$, której zawartość jest tożsama z pierwszymi trzema znakami zmiennej A$.

*) RIGHT$(tekst,argument) - tworzy łańcuch znakowy w sposób podobny jak funkcja LEFT$, z tym, że znaki pobierane są od prawej "strony" (czyli od końca do początku), np.:

```
10A$=''abcdef''
20 B$=RIGHT$(A$,3)
40 PRINT B$

run
def
```

*)MID$(tekst,argument1,argument2) - funkcja, dzięki której możliwe jest utworzenie łańcucha znakowego, będącego fragmentem podanego tekstu (lub zmiennej tekstowej) począwszy od znaku o numerze podanym jako Argument1 i długości podanej w Argumencie2, np.:

```
10B$=MID$(``abcdefghij'',4,3)
20 PRINT B$

run
def
```

Odpowiedzią jest "def", czyli trzyznakowy łańcuch, który pobrano począwszy od 4 znaku w tekscie "abcdefghij", czyli od litery "d". Funkcję MID$ można jeszcze wykorzystać z pominięciem Argumentu2, co spowoduje pobranie łańcucha aż do końca podanego tekstu, np.:

```
10B$=MID$(``abcdefghij'',5)
20 PRINT B$

run
efghij
```

Dla przykładu, korzystając z podanych funkcji, możemy napisać krótki programi, który będzie nam odczytywał czas ze zmiennej systemowej TI$ (patrz: artykuł o zegarze dla C64):

```
10 CZAS$=TI$
20 G$=LEFT$(CZAS$,2)
30 M$=MID$(CZAS$,3,2)
40 S$=RIGHT$(CZAS$,2)
50 PRINT G$;'' godz.
``;M$;''minut``;S$;''
sekund''
```

Na koniec mała propozycja: spróbujcie napisać krótki programik, który kolejno będzie pobierał dane o godzinie, minutach i sekundach używając instrukcji INPUT (patrz: poprzedni odcinek) i w efekcie ustawi zmienną TI$ na żądaną godzinę. Trochę eksperymentów powinno wyjaśnić resztę.

Paweł Sołtysiński

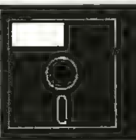
...NEWS...

(c.d. ze strony 6)

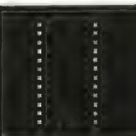
64

Wspominane już dzisiaj demo "More Than NOPs" grupy z Katowic zasługuje moim zdaniem na nieco więcej uznania niż zostało to zawarte w wypowiedzi kolegi "BRUSHa". Demo napisane jest na poziomie światowym i nawet brak rewelacyjnej grafiki nie jest w stanie ująć sporo z bardzo dobrego wrażenia jakie ono sprawia. Bardzo wiele ciekawych pomysłów ze swoistą "listą przebojów" włącznie oraz jakość kodu nie pozostawia wiele do życzenia. Fakt, że demo zostało napisane w Polsce (a na to wygląda) nie polegało zaś na "wycięciu" procedur z produktów grup zachodnich, jest bardzo miłą niespodzianką. To, że jest to debiut autorów dodaje sprawie tylko dodatkowego posmaku i spędza sen z powiek "weteranom sceny".





Na Amigę warto obejrzeć bardzo dobre demo grup Silents/ Crionics z muzyką Jesper Kyd'a o nazwie HARDWIRED.



Słownik skrótów
(część 2)

Witajcie w drugim odcinku naszego słownika skrótów. Dzisiaj postaram się omówić wszystkie terminy związane z trikami używajacymi rejestru $D011. Rejestr ten tylko z pozoru służy jedynie do sterowania przesuwem pionowym ekranu i zmiany trybu graficznego. Podczas wielu eksperymentów okazało się, że wstawiając do niego odpowiednie wartości w odpowiednich liniach (niebędę zagłębiał się w szczegóły techniczne, gdyż jest to materiał na zupełnie inny artykuł) można osiągnać wiele ciekawych efektów. Oto ich kompletna lista:

FLD,VSP,TECH-TECH - triki służące do poruszania ekranu w różne kierunki (omówione dokładniej w poprzednim odcinku).

F.P.D. (flexible pixel distance) - technika pozwalajaca na dowolne regulowanie odstępu między kolejno wyświetalnymi liniami rastrowymi. Pozwala to na efektownie wyglądające "rostrzelenie" linijek obrazka.

F.P.P. (flexible pixel position) - dzięki temu trikowi możliwe jest wyświetlenie dowolnej linii rastrowej w dowolnym miejscu ekranu (tak jak za pomocą Display List na AMIDZE).

F.L.I. (flexible line interpretation) - efekt ten zrewolucjonizował grafikę na C64. Dzieki niemu możliwe stało się uzyskanie w kolorowej grafice wysokiej rozdzielczości 16 różnych kolorow w każdym polu4x8 punktów. Dzięki niemu zniknęły kłopoty z cieniowaniem, które mieli graficy pracujący dotychczas w zwyczajnym multi-kolorze.

A.F.L.I. (advanced F.L.I.) jak wyżej ale dla trybu najwyższej rozdzielczości (16 kolorów w każdym polu 8x8 punktów).

E.C.I. (enhanced color interpretation) jest to ciekawy sposób na otrzymanie 130 kolorów na C64. Polega on na umieszczaniu obok siebie na przemian punktów w dwóch różnych kolorach; powstaje wtedy złudzenie 3 koloru, który jest sumą tych barw składowych. Trik ten jest stosowany wraz z A.F.L.I. i pozwala na tworzenie wielokolorowych wzorków. Niestety nie nadaje się do rysowania obrazków.

Ostatnim, a zarazem najtrudniejszym trikiem korzystającym z rejestru $d011 jest tzw. LINE-CRUNCHER. Dzięki tej technice możliwe jest nie tylko opuszczenie ekranu o dowolną ilość linii ale takze podniesienie go do góry. W ten sposób możliwe jest m.in. płynne przesunięcie obrazka w multi-kolorze do góry o więcej niż 8 punktów. W tym odcinku to już koniec. Zapraszam za miesiąc! Pomówimy wtedy troche o SPRITE'ach, SPLIT'ach i wielu innych ciekawych rzeczach.

Krzysztof "BRUSH" Dąbrowski

Amiga - PC
TEST I

Pierwsze z testowanych urządzeń to najnowsza wersja KCS - Power PC Board, drugie to równie świeży produkt firmy Vortex o nazwie ATonce plus. Dla czytelników nie zorientowanych w temacie garść wyjaśnień odnośnie konstrukcji, sposobu instalacji oraz możliwości obydwu kandydatów testowych. Zacznijmy od KCS'a (w skrócie tak go będziemy nazywać).

Produkująca go holenderska firma Kolff Computer Supplies (KCS) znana m. in. z doskonałego niegdyś "Power Cartridge'a" dla C-64 zadziwiła świat nie tyle samym pomysłem stworzenia emulatora "pecetów" dla Amigi co rozwiązaniem problemu komunikacji pomiędzy emulatorem a Amigą .Powszechne uznanie pism fachowych było odpowiedzią na fakt, że karta emulatora komunikuje się z Amigą poprzez... złącze przeznaczone dla kart rozszerzenia pamięci (od spodu obudowy A500). Fakt ten był przez długi czas przedmiotem krytyki z jednej tylko przyczyny: KCS - Power PC Board nie działał z Amigą 2000 ponieważ nie posiada ona takiego złącza. Problem ten rozwiązano w minionym roku dołączając (za dodatkową opłatą) dla posiadaczy A2000 specjalną kartę przejściową pasującą do"slotów" (złącz typu ZORRO) tego modelu. No ale tu od razu nasuwa się pytanie: Co z pamięcią? jeżeli mam włożyć od spodu A500 emulator to gdzie podłączę pamięć? Niestety nigdzie! Aby nie było to poważniejszym zgrzytem, firma zainsta lowała na płytce emulatora od razu 512 KB RAM'u i zegar czasu rzeczywistego a począwszy od wersji trzeciej znajduje się tam całe 1MB.Pamięc ta może być wykorzystywana zarówno ze strony Amigi jak i MS-Dos'u. Ze względu na pojawienie się na rynku Amigi 500plus, która ma już w fabryce zainstalowany zegar na płycie głównej, firma zdecydowała się na wypuszczenie na rynek specjalnej wersji emulatora przeznaczonej dla "plusa".

Różnice polegają na usunięciu układu zegara oraz akumulatorka z płytki a także na przekonfigurowaniu znajdującej się tam pamięci tak aby mogła ona być "widziana" od strony Amigi jako dodatkowy megabajt Chip-RAM'u. Tak! oznacza to, że po zainstalowaniu KCS'a w Amidze 500 plus mamy przy okazji 2MB pamięci typu Chip. Do emulatora dołączana jest niezbyt starannie wykonana (w porównaniu do samego hardware'u) instrukcja oraz dyskietka instalacyjna. Po (prawidłowym!) włożeniu emulatora w "kieszeń" Amigi należy odpowiednio ustawić konfigurację korzystając z zawartych na dyskietce programów. Jeżeli jesteśmy posiadaczami twardego dysku to musimy wybrać odpowiedni "driver" czyli program sterujący odpowiedni dla naszego modelu. Wybór jest dość spory. W wersji 3.50 (którą testowaliśmy) można wybrać spośród 39-ciu różnych modeli kontrolerów twardych dysków. Znajdują się tu driver'y dla praktycznie wszystkich liczących się produktów z tej dziedziny. Drugi program z dyskietki to właściwy program konfiguracyjny. Pozwala na komfortowe ustawienie wszystkich parametrów pracy emulatora a to: rodzaju emulacji karty graficznej, ilości i rodzaju podłączonych parowozów... przepraszam! STACJI dysków. Można ustalić w jakich kolorach ma się nam "zgłaszać" pecet.

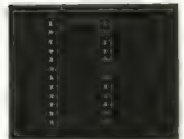
Powtarzalność klawiszy, emulacja myszy pecetowskiej, głośność dźwięku i wiele innych bardziej lub mniej potrzebnych parametrów możemy sobie poustawiać po to tylko aby nasze współżycie z Amigą 500PC było najprzyjemniejsze. Dla leniwych (to znaczy tych co niechcieli się nauczyć angielskiego) zainstalowano w programie możliwość wyboru jeszcze czterech innych języków do komunikacji z użytkownikiem... Ale nie łudźmy się! O polskim nikt nie pomyślał. Jest natomiast niderlandzki, niemiecki, francuski i portugalski. Po ustawieniu wszystkiego co trzeba i wybraniu opcji "SAVE+EXIT", która zapisuje na dysku aktualną konfigurację (pojawia się przy tym sympatyczna animacja), możemy przejść do pierwszego poważniejszego testu tj. "Czy to w ogóle działa?" Uruchamiamy (przy użyciu myszki) właściwy program "Power PC Board". Znika obraz i... nic! Okazuje się, że dla oszczędności miejsca na dysku program jest "zakranczowany" tj. skompresowany (właściwie to nie wiadomo jakiego słowa w końcu na określenie tego procesu używać) Turbo-imploder'em no i trzeba dobrą chwilkę poczekać na dekompresję... JEST! To już chyba to! Po przetestowaniu pamięci zgłasza się elegancka ramka w której (na wzór dzisiejszych AT) przedstawia się nam nasz nowy pecet. No tak! Mały problem! Komputer prosi o włożenie dysku z systemem operacyjnym. Jeżeli nie byliśmy skąpi i kupiliśmy pełną wersję KCS'a tzn. wyposażoną również w oryginalny MS-Dos (4.01) i GW-BASIC na dyskietkach to problemu nie ma. W innym (bardziej ekonomicznym) przypadku pozostaje nam ukraść... tj. Tfu! Chciałem powiedzieć: pożyczyć system od jakiegoś znajomego z IBM'em lub czymś podobnym. Wkładamy dyskietkę i po chwili pojawia się znaczek "A>" jest to oznaką tego, że nasza Amiga pracuje już teraz pod kontrolą zupełnie innego systemu operacyjnego niż przewidzieli dla niej twórcy. Ale nas w tym momencie najbardziej zaczyna interesować JAK (dobrze) ona to robi.

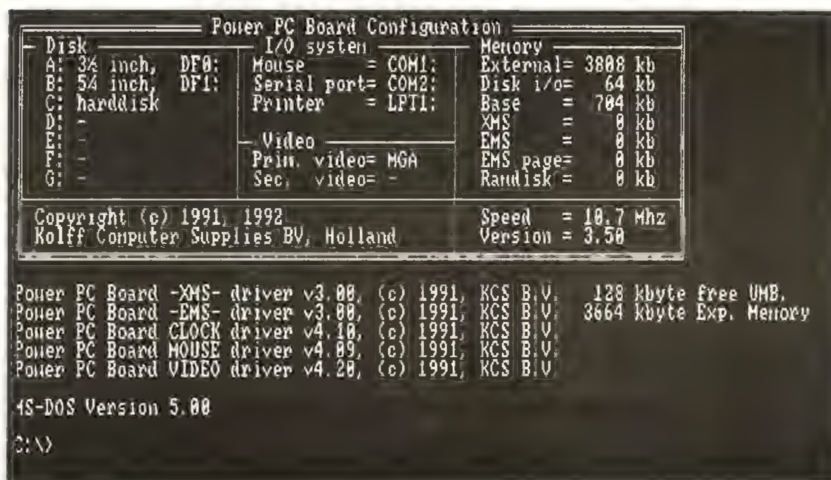
> Często otrzymujemy wiele pytań na temat emulatorów IBM'a przeznaczonych do Amigi. Aby wyjaśnić wiele wątpliwości postanowiliśmy wziąć dwa najbardziej popularne z nich "pod lupę".

Szybko instalujemy twardy dysk (komenda FDISK) kopiujemy nań wszystko co będzie nas interesować i zaczynamy! XTree Pro Gold - działa bez zarzutu. Przy jego pomocy robimy jako taki porządek na "twardym". Norton Commander - również bez problemów. Przechodzimy do programów testujących. System Info Norton'a pokazuje nam CI (współczynnik szybkości wykonywania obliczeń) równe 5.1 natomiast miłą niespodziankę sprawia nam DI (współczynnik szybkości pracy twardego dysku) równy ni mniej ni więcej tylko 16.8! Dla porównania stojący obok "prawdziwy" pecet typu 386-33 MHz z szybkim twardym dyskiem 80MB osiągnął "tylko" 16.5! Pozostałe programy testowe podają podobne rezultaty. Bardzo duża szybkość pracy twardego dysku i stosunkowo niewielka (2 do 3 razy więcej niż PC/XT) szybkość obliczeń. Co z innymi poważniejszymi rzeczami? Sławetna "Ventura" pracująca pod "GEM'em" (tj. jedną z graficznych "nakładek" na system MS-Dos) owszem uruchamia się nawet z tzw. Proffesional Extension (pod warunkiem posiadania odpowiedniej ilości pamięci w Amidze) lecz szybkość a właściwie jej brak przynajmniej w trybie VGA powoduje, że praca staje się niemożliwa. Zdecydowanie lepiej jest w pozostałych trybach. Windows to drugie magiczne hasło z kręgu fanów komputerów "ajbiemopodobnych". Również działa, ale również na tyle wolno, że na dłuższą metę praca z tą nakładką staje się męcząca. Windows z resztą sam w sobie jest straszliwe wolny i bez 386-ki lub specjalnej karty przyspieszającej nawet na prawdziwym pececie nie jest zbyt przyjemny w pracy. Co jeszcze trzeba sprawdzić? Oczywiście osławiony Chi-Writer (sławny chyba jedynie dlatego, że ktoś zrobił mu korzystając z dołączonego edytora polskie znaki w trybie graficznym. A tak na marginesie to lepsze rezultaty w wydruku przynosi skorzystanie z PrintFox'a czy Font-Master'a na C-64...). Chi-Writer działa bez zastrzeżeń zarówno w trybie CGA jak i Hercules (innych driverów w dostępnej mi "kradzionej" wersji nie było). Jedynym mankamentem w trybie Hercules jest konieczność korzystania z tzw. funkcji Overscan Amigi. Chodzi o to, że karta HGC (Hercules Graphics Card) posiada rozdzielczość poziomą przekraczającą 700 punktów i owszem Amiga potrafi korzystając z w/w funkcji obsłużyć tak szeroki ekran, ale może on się nie zmieścić na monitorze! Istnieje wprawdzie możliwość "przesuwania" ekranu lecz znacznie lepszym rozwią-









Ekranik KCS'a...

zaniem wydaje się być skorzystanie (o ile to możliwe) z innego trybu graficznego bardziej trzymającego się standardów rozdzielczości. Następnym edytorem który postanowiliśmy "wziąć na warsztat" był oryginalny TAG w wersji 2.01. Działa bez zastrzeżeń! Dowodem tego jest niniejszy tekst napisany w całości edytorem TAG na Amidze (oczywiście wyposażonej w KCS'a). Znowu jednak możemy wybrać wszystkie tryby graficzne oprócz VGA ponieważ w tym trybie edytor staje się zbyt powolny. Gry! Tu nie spodziewaliśmy się właściwie żadnych pozytywnych rezultatów. Dużym zdziwieniem było to, że udało nam się uruchomić (a nawet pograć) w niektóre z nich. Nie należy jednak nastawiać się na żadne rewelacje. Gry są zazwyczaj programowane w taki sposób, który skutecznie uniemożliwia jakąkolwiek emulację. Co jeszcze? Do sympatycznych ciekawostek należy zaliczyć kilka elementów, których nie posiadają oryginalne pecety a które możemy sobie zafundować korzystając z KCS'a. A to: zegar oraz datę u dołu ekranu działające niezależnie od tego co dzieje się na ekranie zasadniczym. Tzw. "track-display" czyli wyświetlacz aktualnie obsługiwanej (zapis lub odczyt) ścieżki na każdej z dyskietek. Płynna regulacja prędkości procesora od 20% do 100%. Bardzo przydatne do gier (jeżeli działają) gdzie możemy sobie ustalić z jaką prędkością będziemy grać. I wiele innych drobiazgów niedostępnych na prawdziwym komputerze typu PC.

Drugi kandydat testowy to Vortex ATonce plus. Co to takiego? Znany na rynku od dłuższego czasu ATonce został w wersji "plus" dosyć znacznie przekonstruowany. Poprawiono przede wszystkim elementy, które były głównym przedmiotem krytyki w odniesieniu do pierwotnego ATonce'a (sprzedawanego z resztą w dalszym ciągu pod nazwą ATonce classic). Przede wszystkim szybkość. Nowa wersja posiada procesor 80C286 taktowany z częstotliwością 16 MHz. Producent zapewnia, że emulator ten osiąga współczynnik Norton'a równy 16.2. Potwierdza się to również w naszych testach. Oprócz tego dodano pamięć. Na płycie emulatora znajduje się 512KB RAM'u. W poprzedniej wersji ATonce musiał korzystać z pamięci Amigi. Niestety pamięć emulatora jest dostępna tylko dla niego tzn. nie jest "widziana" przez Amigę i nie może być wykorzystana jako np. dodatkowe pół megabajta Fast-RAM'u. Dodano również gniazdo dla koprocesora arytmetycznego. Wstawienie tam układu typu 80c287 powinno znacznie zwiększyć szybkość wykonywania obliczeń arytmetycznych (bardzo istotne w przypadku korzystania z programów np. typu CAD). Instalacja ATonce'a nie jest już tak prosta jak w przypadku KCS'a. Musimy zacząć od rozkręcenia obudowy. Po zdjęciu klawiatury i blachy ekranującej, odszukujemy nasz (Amigowy) mikroprocesor. Nie powinno to nastręczyć zbytnich trudności jako, że jest on po prostu największym układem na płycie. Wyciągamy go następnie wkładamy w podstawkę po nim drugą podstawkę dołączoną do emulatora (chodzi o to aby podwyższyć mocowanie) i dopiero w nią wkładamy dokładnie i ostrożnie płytkę samego ATonce'a. Na tej płytce znajduje się gniazdo w które należy włożyć dopiero co wyjętą "Motorolę" i dopiero wtedy możemy zacząć składać z powrotem Amigę. O ile płytka KCS'a pozostawiała wrażenie bardzo dokładnie wykonanej o tyle w przypadku produktu firmy Vortex jest to już najwyższej klasy technologia. Wykonanie jest bardziej niż zegarmistrzowskie. Montaż powierzchniowy wszystkich niemalże elementów nie



...i ekranik ATonce'a

RODZAJ EMULACJI	TYP	KOLORY	FORMAT
CGA-MODE 0	TEXT	16	40*25 ZNAKÓW
CGA-MODE 1	TEXT	16	40*25 ZNAKÓW
CGA-MODE 2	TEXT	16	80*25 ZNAKÓW
CGA-MODE 3	TEXT	16	80*25 ZNAKÓW
CGA-MODE 4	GRAFIKA	4	320*200 PUNKTÓW
CGA-MODE 5	GRAFIKA	4	320*200 PUNKTÓW
CGA-MODE 6	GRAFIKA	2	640*200 PUNKTÓW
MDA	TEXT	2	80*25 ZNAKÓW
HERCULES	GRAFIKA	2	720*348 PUNKTÓW
TOSHIBA 3100	GRAFIKA	2	640*400 PUNKTÓW
OLIVETTI	GRAFIKA	2	640*350 PUNKTÓW
EGA-MONO	GRAFIKA	2	640*350 PUNKTÓW
VGA-MONO	GRAFIKA	2	640*480 PUNKTÓW

Graficzne możliwości Vortex'a...

należy już do rzadkości, ale rzadko spotyka się tak precyzyjne wykonanie w produktach przeznaczonych na rynek komputerów "domowych". Po tych zachwytach nad hardware'm przejdźmy na chwilę do oprogramowania. Dołączone na dwóch dyskietkach (jedna w formacie Amiga-Dos, druga MS-Dos) zawiera oprócz samego programu obsługującego emulator również program konfiguracyjny oraz instalacyjny. Program konfiguracyjny mimo iż nie tak elegancki i dopracowany jak w przypadku KCS'a, spełnia jednak swoje zadanie. ATonce (w przeciwieństwie do Power Board'a) nie przejmuje w momencie uruchomienia kontroli nad całą Amigą lecz działa w oparciu o jej system operacyjny. Pozwala to na pełny "multitasking" czyli możemy jednocześnie pracować na Amidze i AT. Jak trzeba to możemy nawet obsunąć ekran "pecetowski" tak jak każdy inny ekran Amigi. Ma to niestety również pewne wady. Na przykład dyskietki zapisane w formacie MS-Dos po włożeniu do drive'u (czytaj: napędu, stacji) muszą zostać zweryfikowane przez Amiga-Dos. Przedłuża to znacznie czas pomiędzy włożeniem dyskietki a momentem w którym możemy już z niej korzystać. Oprócz tego również inne operacje we/wy wykonywane przez Amigę znacznie spowalniają pracę emulatora tj. przede wszystkim obsługę ekranu gdyż procesor Intel działa niezależnie i system operacyjny Amigi nie ma na niego wpływu. Aby temu zaradzić można w programie konfiguracyjnym ustawić opcje "Stop Amiga Boot" i "ATonce only". Funkcje te zatrzymują boot'owanie Amigi zaraz po uruchomieniu ATonce'a oraz uniemożliwiają przełączanie pomiędzy systemami. Aby uaktywnić nasz emulator musimy wybrać odpowiednią ikonkę i wykonać na niej odpowiednią (dwukrotną) operację (dla niedomyślnych wyjaśniam, że chodzi o: tupnięcie, mlaśnięcie, kliknięcie lub coś w tym rodzaju). Gdy zrobimy to pierwszy raz to będziemy mieli wrażenie, że coś jest nie tak. Zamiast pojawienia się ekranu IBM'a Amiga wykonuje normalny reset! Lecz już po chwili wszystko się wyjaśnia pojawia się odpowiedni ekran i po przetestowaniu pamięci prosi o włożenie dyskietki systemowej. Jeżeli posiadamy coś twardego (np. dysk) to również mamy możliwość skorzystania z niego w trybie AT. Po zainstalowaniu wszystkiego co trzeba możemy wziąć się do testów. Testy szybkości: CI 16.2, DI niestety tylko 6.6 a więc na poziomie zwykłej nieco podstarzałej 286-ki. Co rzuca się w oczy (w porównaniu z KCS'em) to fakt, że o ile szybkość wykonywania obliczeń jest istotnie większa o tyle obsługa ekranu oraz dysków jest w przypadku ATonce'a zdecydowanie wolniejsza. Szybkość emulacji trybu tekstowego (nie występującego w ogóle w Amidze) jest przy użyciu KCS'a mniej więcej na poziomie standardowego AT, natomiast ATonce sięga (w tej dziedzinie) nie wyżej niż do poziomu XT. Przyjemną niespodzianką natomiast była obsługa Windows'ów - szybka na tyle aby umożliwić pracę bez poważniejszych ataków nerwowych. Ventury w trybie VGA nie udało się do końca uruchomić. Norton Commander i Xtree - bez zarzutów. Co należałoby powiedzieć jako podsumowanie aktualnego stanu jakościowego emulatorów PC na Amigę? Przede wszystkim: To działa! Naprawdę! Chociaż nie zawsze tak jak by się chciało, to jednak fakt pozostaje faktem: w obudowie od "pięćsetki" może zmieścić się również komputer typu IBM PC. Kompatybilność mimo tego iż nie zawsze stuprocentowa choćby ze względu na konieczność emulowania ekranu PC poprzez ekran Amigi, jest jednak stosunkowo wysoka. CZY oraz KTÓRY emulator należałoby komukolwiek doradzić? Prawdę mówiąc obydwa są w pewnym stopniu uzupełnieniem dla siebie i różnie się sprawdzają w różnych zastosowaniach. Jeżeli potrzebny jest komputer typu AT i planujemy korzystać z systemu Windows to możemy od razu zapomnieć o KCS'ie. Jeżeli natomiast zamierzamy korzystać z prostszych programów (chociażby wspomniany już Chi-Writer lub Tag) ewentualnie programować bazy danych w DBase, Clipper itp. to taniej wychodzi zainstalowanie właśnie tego emulatora. Nie bez znaczenia w tym przypadku pozostaje również dużo większa szybkość obsługi ekranu i dysków oraz możliwość korzystania z pamięci KCS'a również pod kontrolą Amigi. Ewentualne posiadane już rozszerzenie pamięci można np. sprzedać co dodatkowo obniży poniesiony przez nas koszt.

zdjęcie strona 32

SD!

...a to już KCS'a

RODZAJ EMULACJI	TRYB	KOLORY MAX	ROZDZIELCZOŚĆ
CGA	GRAFIKA, TEXT	16	620*200
MGA	GRAFIKA, TEXT	2	720*348
EGA	GRAFIKA, TEXT	16	640*350
VGA	GRAFIKA, TEXT	16	640*480



Cześć! Cześć! Cześć!

Chciałbym szanowny Kebabie zwrócić się do Was o pomoc. Swój problem przedstawię w punktach.

1. Czy A500+ to A3000 w obudowie A500?

2. Jaką A500 czy A500+ warto kupić? Proszę o podanie różnic tych komputerów.

3. Ile kosztuje przełącznik Kickstart'a z 1.3 na 2.0 lub odwrotnie?

4. Chciałbym nauczyć się programować (gry, programy edukacyjne, użytkowe). Jakich mógłbym używać języków programowania do odpowiednich programów?

5. Czy A500 nie jest już "starym" komputerem do IBM-a lub Atari TT?

6. Jakie są najlepsze drukarki, modemy, skanery na Amigę, oraz programy obsługujące w/w osprzęty.

7. Czy narysowany obrazek np. za pomocą DPainta III można nagrać na dysk, a później wyświetlić go automatycznie wkładając dysk do stacji?

8. Czy pisząc program zamiast dopisywać muzykę można ją wyciąć z innego programu? Proszę wymienić niezbędne do tego programy.

Marcin z Bydgoszczy.

1. Nie! A500+ to A500+ w obudowie od A500+! Ale bez żartów. Cechy wspólne dla A500+ i A3000 to nowe tryby graficzne (Denise), do 2MB Chip-RAM (Agnus) oraz system operacyjny OS-2.0 (Kickstart, Workbench). Cechy odróżniające oba modele to w A3000 procesor 68030 (32-bitowa architektura), Flicker-Fixer, Twardy Dysk itd.

2. Dokładny test Amigi 500+ zamieściliśmy w pierwszym numerze Kebaba. W listach do redakcji (2-3/92) pisaliśmy jeszcze więcej na ten temat.

3. Sam przełącznik nie jest bardzo drogi (ok. 150-200 tys. zł.), ale trzeba jeszcze mieć układ ROM z Kickstart'em który jest znacznie droższy (60-200 DM zależnie od wersji).

4. Wybór języka programowania istotnie zależy od tego jakie zadania ma realizować przyszły program. Zatem przed napisaniem jakiegokolwiek programu należy się dobrze zastanowić jaki będzie jego krąg odbiorców, jakim czasem dysponujemy na jego wykonanie, czy ma być na tyle szybki, że zachodzi konieczność ucieczki do języka maszynowego, i dopiero po rozważeniu wszystkich argumentów podjąć decyzję o wyborze języka. A oto lista języków programowania i ich potencjalnych zastosowań:

AMOS: programy edukacyjne, matematyczne, krótkie gry;

ARexx: komunikacja międzyprogramowa;

Assembler: gry, dema, użytki, procedury krytyczne czasowo;

Basic: krótkie obliczenia matematyczne, drobne programiki;

C: użytki, programy matematyczne;

Pascal: programy matematyczne.

5. A500 na pewno jest starym komputerem w porównaniu do Atari TT. Została "wypuszczona" na rynek w roku 1986! Atari TT natomiast dopiero w latach dziewięćdziesiątych. Co do IBMa to nie sprecyzowałeś o jaki model Ci chodzi więc trudno cokolwiek szacować. Nie wiem również jak traktować cudzysłów przy słowie "starym". Moim zdaniem sprzęt powinien być kupowany z myślą o tym jakie zadania ma wykonywać a nie czy jest "modny", "stary", "nowy" itd.

6. Podobnie jak wybór języka programowania, tak i tym bardziej wybór osprzętu zależy od szczegółowych oczekiwań indywidualnych użytkowników. Obecnie na rynku znajduje się cała gama peryferiów do Amigi w ogromnej rozpiętości cenowej. Oczywiście nie można pokusić się o stwierdzenie, że najdroższe są najlepsze, a nawet nie można powiedzieć, że wśród drukarek najlepsze są ostatnio bardzo modne "laserówki"! Bo jeśli ktoś chce na przykład drukować rachunki w magazynie, najtrafniejszym wyborem wydaje się zwykła 9-cio igłówka i kupienie drukarki laserowej z myślą o tylko takim zastosowaniu wydaje się być po prostu zwykłym snobizmem. Powtarzam więc, że każdy nabywca jakiegokolwiek sprzetu, nawet niekoniecznie komputerowego powinien zawczasu wiedzieć do czego będzie go wykorzystywał.

7. Oczywiście gotowy obrazek wykonany na dowolnym programie graficznym można później wyświetlić i najlepszym sposobem wydaje się być skorzystanie z programu PPShow. Aby cały proces przebiegł automatycznie należy w pliku "S:startup-sequence" umieścić linię:

PPShow NazwaObrazka

Teraz podczas każdego boot'owania (czyli najprościej mówiąc: wkładania dyskietki podczas, gdy na ekranie znajduje się rączka z dyskietką) dysku z zapisanym uprzednio obrazkiem pod nazwą "NazwaObrazka", nasze dzieło zostanie automatycznie wyświetlone. Oczywiście na dyskietce powinien znajdować się również program PPShow iw katalogu LIBS biblioteka "powerpacker.library".

8. Najpopularniejszymi programami do wycinania modułów muzycznych są: David Whittaker Ripper, Cosmo Ripper, Flash Ripper, Multi Ripper v3.0. Również cartridge Action-Replay pozwala na przeprowadzenie tego zabiegu i wydaje się być w tym wypadku najlepszym rozwiązaniem, gdyż użytkownik posiada możliwość wglądu do pamięci, podczas wykonywania dema lub gry z których chcemy wyciać muzykę, tj. wtedy, gdy na pewno znajduje się ona w pamięci. Wejście do wspomnianych wyżej programów możliwe jest dopiero po wykonaniu resetu komputera, co w razie pozostawienia ewentualnych zabezpieczeń przez autorów gry (dema), lub zajęcia przez nią obszaru pamięci zarezerwowanego dla systemu operacyjnego może prowadzić do wyzerowania pamięci RAM, a co za tym idzie zniszczenia danych muzyki. Na zakończenie wspomnę jeszcze, że nie wszystkie muzyki można wyciągnąć za pomocą innych programów do tego przeznaczonych i czasem niezbędne jest posłużenie się monitorem pamięci i wiedzą tajemną.

Jestem użytkownikiem C-64 od niedawna. I od początku (gdy zobaczyłem demo grupy QUARTET "Video Box") trapi mnie problem - jak "zroblć" własne demo. Główkowałem cały miesiąc, próbując wymyślić coś w Basicu. Niestety okazało się to niemożliwe dla mnie. Więc w końcu zdecydowałem się napisać do was. Proszę was, pomóżcie mi !!! Jeśliby było to możliwe to chociaż proszę o odpowiedź na kilka pytań, które przygotowałem. Myśle, że odpowiedź na nie nie sprawi wam takiego kłopotu jak mi.

1. Gdzie można nabyć LOGO-EDITOR?

2. Jak POLONUS robl te "zadymiarskie" (chyba - przyp. red.) napisy w demach?

3. Jak szybko nauczyć się "obsługi" rastra?

4. Czy można zrobić "scrolla" w Basicu i umieścić go w dowolnym miejscu ekranu?

5. Jak zrobić jakiś zestaw znaków lub wyciąć go z jakiegoś dema

6. Jak można zrobić muzyczkę w assemblerze i użyć ją w demie?

7. Czy obraki zrobione programem "ART STUDIO" lub "MICRO ILLUSTRATOR" można wykorzystać w demie?

8. Jak wpakować własne demo przed: grę, program lub dołączyć do innego demo?

9. Jestem posiadaczem programu do układania własnej gry. Niestety jednak ułożoną grę można zgrać tylko w "normalu" i załadować po uprzednim wgraniu tego programu. Jak to zmienić?

Z poszanowaniem - >>ICE-MAN<<
Radosław Waśkiewicz

UUUFFF!!!

Drogi ICE-MAN'ie po raz kolejny czytam twoje "kilka" pytań i nieodparcie nasuwa mi się przekonanie, że aby odpowiedzieć na nie

wyczerpująco, musielibyśmy wydać całą książkę dotyczącą programowania C-64. Mimo tego ponieważ pytania twoje są jakby zbiorem problemów o które bardzo często pytają nasi Czytelnicy, spróbujemy jakoś się do nich ustosunkować.

1. Sądzę, że chodzi Ci o "Logo Editor" grupy CENTAURI. Jest on programem typu Public Domain i powinien być dostępny u większości handlarzy giełdowych. Lepszym programem tego typu jest "Multi-Drawer" wydany przez niemiecką firmę CP Verlag. Jeżeli wszystko pójdzie poprawnie, najnowszą wersję tego programu będzie można wkrótce kupić u nas.

2. Nie znając definicji "zadymiarskiego" napisu, trudno mi jest stwierdzić o jakie napisy Ci chodzi. Przypuszczam, że miałeś na myśli wszelkiego rodzaju scrolle (w górę, dół, lewo, prawo, po skosie, po sinusoidzie itp.). Odpowiedź na to pytanie znajdziesz śledząc kolejne odcinki kursu assemblera oraz rozpoczynający się wkrótce cykl "Procedury trickowe".

3. Odpowiedź, patrz punkt drugi.

4. Odpowiedź, patrz KEBAB 4'92 strona 22.

5. Kolejne pytanie na które odpowiemy w naszym cyklu artykułów poświęconych programowaniu w assemblerze. Dla nie zainteresowanych tematem "jak to działa" lecz tylko: "jak to zrobić najprościej" polecamy dowolny edytor znaków (ang.: character editor) jakich sporo znajduje się w obiegu. Programy te pozwalają na skonstruowanie dowolnego zestawu znaków i korzystanie z niego we własnych programach. Zwracamy jednak uwagę na drugą część pytania. Zrobić własny zestaw znaków to jest OK! Ale "wyciąć" go, to już inna sprawa. Jeżeli program z którego coś wycinamy jest typu PD (Public Domain) to możemy wycinać do woli. Niemniej jednak jeżeli nie chcemy się narazić na antypatię ze strony autorów, należy zaznaczyć w swoim programie skąd zestaw znaków (jak również procedura, obrazek, muzyka) naprawdę pochodzi. Przypomina mi się tu przykład niemieckiej grupy "Baboons". Grupa ta wypuściła serię programów demonstracyjnych chwaląc się w każdym z nich jakie wspaniałe procedury obsługi dysku zostały przez nich napisane. Oczywiście prawda szybko wyszła na jaw a prawdziwy autor procedur zyskał tylko jeszcze większą sławę (dla zainteresowanych - RADWAR). Wspomniana wyżej grupa bardzo szybko przestała istnieć. No, ale wracając do pytania. Nie ma jednoznacznej metody "wycinania" zestawów znaków. Mając odpowiednią wiedzę na temat działania komputera można jednak zawsze sobie poradzić. Mniej zaawansowani "hackerzy" powinni skorzystać z programów typu "Charset-Ripper" (wkrótce w KEBAB'ie).

6. W assemblerze można napisać procedurę odtwarzającą muzyczkę, natomiast samo komponowanie przy użyciu bitów i bajtów byłoby na tyle uciążliwe, że nie sądzę żeby ktoś podejmował się takiej pracy tym bardziej, że istnieją gotowe edytory umożliwiające komponowanie muzyki i zawierające w sobie od razu gotowe procedury odtwarzające (napisane w assemblerze). Pierwszym tego typu edytorem, posiadającym do dziś swoich zwolenników, był Sound Monitor autorstwa Chrisa Huelsbecka, wydany w 1986 roku przez wydawnictwo Markt & Technik. Po nim przyszły następne: Future Composer, Ro-Muzak, Music Assembler oraz najlepszy chyba aktualnie na rynku - The Music Mixer. Programy te umożliwiają nie tylko skomponowanie i posłuchanie muzyczki, ale również (dzięki samodzielnym procedurom odtwarzającym) wykorzystanie kompozycji we własnych programach.

7. TAK! Najlepiej jeżeli będzie to tzw. "Advanced Art Studio", umożliwiające malowanie w trybie "multicolor".

8. Ojej! Znowu temat na duży artykuł. Najlepiej "wpakować" własne demo nie przed, a ZA grę... itp. To znaczy napisać je tak aby zajmowało i wykorzystywało obszar pamięci nie zajęty przez (nie uruchomioną) grę. Wymaga to dobrej znajomości działania komputera oraz (najlepiej) języka maszynowego. Zatem: Cierpliwości!

9. Znam tylko jeden program działający w sposób opisany w Twoim liście, lecz nie radziłbym Ci nic w nim zmieniać z przyczyn podobnych jak w punkcie ósmym. A gdy nabierzesz już wprawy w posługiwaniu się językiem maszynowym, prawdopodobnie stwierdzisz, że nie jest Ci ten program już potrzebny bo i bez niego potrafisz zaprogramować tego typu grę.

Prawdopodobnie po przeczytaniu powyższych odpowiedzi wielu Czytelników stwierdzi, że nic lub bardzo niewiele z nich wynika. I słusznie! Bowiem tak jak nie da się zjeść jabłka rozpoczynając od zawartych w samym środku pestek, tak również nie można "nauczyć się obsługi rastra" czy "zrobić muzyczki w Asemblerze" nie znając języka asemblera i rejestrów sprzętowych VIC'a odpowiedzialnych za obsługę "rastra". Większość poruszonych tematów zostanie szczegółowo omówiona na łamach KEBAB'a w artykułach ich właśnie dotyczących. Niecierpliwym polecam książki w języku angielskim: "C64 - Machine Language for absolute beginners" (COMPUTE! publications), "Commodore 64 Programmers Reference Guide" (Commodore) i niemieckim: "Maschinensprache Buch", "Maschinensprache Buch fuer fortgeschrittene", "64 Intern" - wszystkie wydane przez wydawnictwo "Data Becker". Z polskich pozycji najbardziej godną polecenia wydaje się być książka p. Frelka - "Commodore 64".

Jestem posiadaczem COMMODORE 64 od trzech miesięcy. BSIC V2 znam prawie na wylot gdyż interesowałem się informatyką od 5-ciu lat. Piszę odnośnie artykułu w pierwszym numerze KEBABa dotyczącego cartridge-ów a mianowicie posiadam Black Box V4.0 i jestem nie zadowolony gdyż po wczytaniu MONTEZUMA'S REVANGE, moduł blokuje komputer a po chwili zgłasza się system kartridża jak gdyby nigdy nic z wyczyszczoną całkowicie pamięcią. Moi koledzy, ktÓrzy posiadają wersję 4.0 mają takie same problemy. Domyśliłem się, że jest to moduł polski gdyż ma polskie napisy pozatym polskie napisy widnieją na płytce drukowanej a w skład modułu wchodzi pamięć EPROM oczywiście z zamazanym typem a producentem EPROMU jest znana firma TEXAS INSTRUMENTS oraz polski układ TTL UCY7438 bodajże. Piszę o tym ponieważ interesuję się elektroniką poza tym interesuje mnie sprawa braku adresu producenta CARTRIDGEa. Prawdopodobnie jest to moduł piracki z obrzydliwą instrukcją obsługi tak jak fałszywe FINAL czy AKTION.

Z uszanowaniem:

Krzysztof Stefanowicz z Ładzina





Drogi czytelniku, z naszych doświadczeń wynikają podobne wnioski (patrz również artykuł Cartridge?...) dotyczące cartridge'ów "Black Box" a w zczególności V4.0. Dokładniejsza analiza programu zawartego w cartridge'u ujawniła nam, że autorzy dzięki zbyt swobodnemu wykorzystywaniu wektorów systemowych C-64 do własnych celów (obsługi procedur) doprowadzili do znacznej niekompatybilności ze sporą ilością tytułów programowych. Niestety bez przeprogramowania całego systemu cartridge'a nie da się nic poprawić.

Jestem użytkownikiem C-64. Piszę do was z prośbą ponieważ mam grę "LAST NINJA II". Nie umiem przejść: The Sewers, The Office, Final Battle. Prosiłbym was o podpowiedzenie mi jak mam przejść te poziomy.

Grzegorz Kamoń

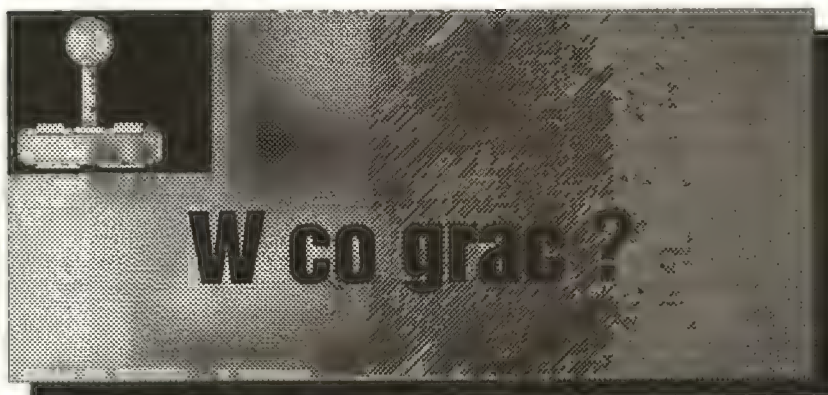
Niestety drogi Czytelniku nie możemy podpowiedzieć Ci "jak masz przejść te poziomy". Wymagałoby to co najmniej dodatkowego artykułu dotyczącego wspomnianej gry. Niemniej jednak może ktoś z Czytelników chciałby pomóc Grzegorzowi w rozwiązaniu problemu. Czekamy na ewentualne listy w tej sprawie.

Piszę do was w sprawie testu Amigi 500 plus (Kebab nr 1). Chciałbym kupić sprzęt nie tylko do zabawy ale i do poważniejszych zastosowań. W związku z tym mam pytanie ile procent programów użytkowych "nie chodzi" na tym komputerze? Druga sprawa to cena A500 plus. Pod testem była podana cena ok.5800 tys.zł.Gdzie ta cena obowiązuje? Gdzie można kupić Amigę za taką cenę?

Mikołaj Chorczun



Może najpierw sprawa ceny. Podana pod testem cena obowiązywała w sklepach firmy "HANDWIT" w Grudniu ubiegłego roku. Niestety kolejna zmiana stawek celno-podatkowych oraz zmiany kursu walut zachodnich uczyniły podaną cenę nieaktualną. Co do programów użytkowych to jak wspominaliśmy w teście niewielka ilość z tych które próbowaliśmy uruchomić, odmówiła posłuszeństwa (tj. "chodzenia"). Nie jesteśmy ani my ani chyba nikt inny w stanie powiedzieć dokładnie ILE PROCENT takich programów jest na rynku. Niemniej jednak WSZYSTKIE nowe tj. aktualnie wydawane programy MUSZĄ być dostosowane do systemu operacyjnego OS-2.0 gdyż w przeciwnym wypadku nie mają szans na dobre wyniki sprzedaży. Tak więc w przypadku "poważniejszych" programów posiadacze "plusa" mogą spokojnie patrzeć w przyszłość. Z programów nie działających należy wymienić "Turbo Silver" oraz serię "Tracker'ów".

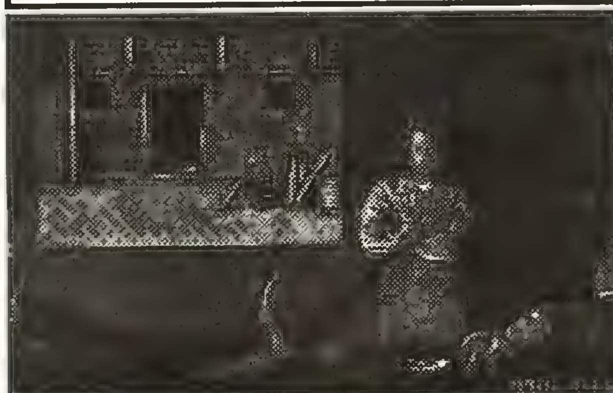


ALCATRAZ

USA, rok 1996. Wyspa Alcatraz jest teraz główną kwaterą wielkiego kartelu narkotykowego prowadzonego przez Miguela Tardieza'a. Ty jako komandos odziału do zadań specjalnych masz spenetrować całą wyspę i uwięzić Tardiez'a. Tuż przed twoim przybyciem wywiad ustalił kilka danych. Na samym początku dostajesz się niezauważony na wyspę za pomocą canoe Pierwsza częśc misji to spenetrowanie budynków i znalezienie tajnych papierów Tardiez'a w których zawarte są dane wszystkich jego współpracowników. W fabryce twoim celem są również zapasy narkotyków i pieniędzy. Musisz znaleźć dwa właściwe pokoje i umieścić w każdym z nich bombę czasową. Drzwi w trzecim budynku są nie do przejścia. Musisz użyć haka i wejść przez dach. W środku trzeba uwięzić Tardiez'a i... Na dostanie się do punktu w którym czeka na ciebie helikopter masz dwie godziny. Potem helikopter odleci bez ciebie. Gra nad którą ciężko zatrzymać się dłużej, chyba że ktoś naprawdę lubi tego rodzaju gry. Po średnio kiepskim intrze z animacją fal w trzech fazach (aż oczy bolą) i wprowadzeniu w scenariusz gry, zaczyna się rąbanina. Nie jest jednak łatwo uśmiercić przeciwników w tej grze.Na początku mamy do dyspozycji tylko własne ręce i dziewięć noży.W momencie rzutu nożem postać naszego komandosa zatrzymuje się, co znacznie utrudnia płynny ruch. Poza tym aby zabić jednego bandytę trzeba mu wsadzić w kichy co najmniej trzy noże. Jest to o tyle denerwujące, że zanim zrobimy mu ostateczne kuku, ten zdąży jeszcze odebrać nam kilka punktów energii.

Trochę lepiej jest z granatami. Trzeba jednak pamiętać, że nasze cele są ruchome. Zanim kozłujący granat raczy wybuchnąć, w międzyczasie może dojść już do rękoczynów. W efekcie trzeba być nie lada wróżką, by przewidzieć ruch naszego przeciwnika. Doskonale natomiast nadają się granaty do popełnienia samobójstwa. Po dalekim wyrzucie i dalszym posuwaniu się w tym samym kierunku, wylądujemy z kupą odłamków w ciele oraz zmniejszoną energią. Jest to jednak bardzo groźna broń gdyż ani nasze ciosy, ani nawet miotacz ognia nie mają szkodliwego wpływu na zdrowie naszego partnera z drugiego ekranu. Granat wprawia go jednak w chwilową zadumę. Zabijani bandyci zostawiają nam w spadku różne rodzaje broni. Miotacz ognia jest dość skuteczny, ale wolny. Karabin jest już całkiem znośny. Jeżeli jednak nasz komandos rozszaleje się nieco, zdobywanie magazynków stanie się jego życiową pasją.

Ten pan wytłumaczy nam wszystko...



Dopatrzeć też można się w tej grze kilku błędów programowych. Na przykład podczas walki wręcz nagle pojawiają się części ciała nie wiadomo czyje i skąd. Jest to trochę rozpraszające, ciężko wyczuć z którą częścią aktualnie walczymy, a która ma ochotę nam przyłożyć.

Ekran podzielony jest na dwie części. W przypadku dwóch graczy jest to sprawa zrozumiała. Ciężko jednak zrozumieć dlaczego pojedynczy gracz nie ma możliwości zrezygnowania z drugiej stojącej połówki. Wiem, że znajdą się geniusze którzy spokojnie poprowadzą obu komandosów jednocześnie (klawisz Enter). Przeciętny śmiertelnik szybko jednak pogubi się nie wiedząc którym komandosem walczyć, a którym zbierać broń. Efekt może być tylko jeden - rachityczny napis "Game Over". Zamiast tego podejrzewam, że ciekawiej było by oglądać na ekranie pełnowymiarową grafikę.

Na szczęście grafika jest również na średnim poziomie, więc nie ma czego żałować. Horyzontalny scroll, kiepska animacja. Czasami wyróżnia się kilka statyczych obrazków na które można nieco dłużej popatrzeć.

Ogólnie gra jest raczej godna polecenia jedynie dla specjalistów, ludzi którzy naprawdę gustują w tego rodzaju grach. Tym bardziej, że gra na poziomie easy (łatwy) jest stosunkowo trudna. Nie wiem co w takim razie będzie się działo na poziomie hard (bardzo trudny). Z założenia wskazana jest gra we dwóch graczy. Przejście poszczególnych etapów staje się wtedy znacznie łatwiejsze, a przyjemność z zabijania ekranowych bandytów przedłuża się o parę cykli zegarowych. Przy odrobinie szczęścia można wtedy dojść do końca poznając całą historię. A do pełnego standardu tego oklepanego scenariuszu brakuje jedynie zakładniczki Tardiez'a, niespotykanej urody dziewicy którą można by uwolnić i żyć z nią długo i szczęśliwie. W sumie muszę się chyba przyznać do jednej rzeczy. Po spędzeniu kilku upojnych chwil przy Altacraz nie wiem już czy ta gra jest taka mama

czy może ja po prostu nie lubię tego rodzaju produktów. Cóż, zawsze jest to rzecz gustu!

McGreg

STORM MASTER

Długo zastanawialiśmy się w gronie redakcyjnym nad wyborem gier do tego numeru. W przeciwienstwie do innych czasopism na naszym rynku w Kebabie bardzo niewiele poswięca się miejsca na gry, wskazówki, cheat'y itd. Powoduje to oczywiście chęć trafienia zamieszczonymi artykułami do jak największego grona czytelników, co wierzcie mi nie jest rzeczą prostą.

Tym razem z pomocą przyszła nam firma Silmarlis wypuszczając na rynek swój ostatni produkt STORM MASTER. Formuła gry Arcade-Strategy powinna zadowolić zarówno fanatyków jojstick'a, jak i domorosłych strategów. Wysokie, szóste miejsce Storm Master'a na liście najpopularniejszych gier w tej kategorii dowodzi, że i tym razem Silmarlis trafiła w dziesiątkę. Przenieśmy się zatem na planetę Urgaa, na malowniczą wyspę Eolia. Tam właśnie z uwagi na przykry wypadek (morderstwo) zostaliśmy wybrani na przewodniczącego miejscowego rządu. A czas to nienajlepszy by piastować taki urząd. Tuż obok naszego lądu znajduje się druga wyspa Sharkaania i nikt już nie pamięta od jak dawna dwa narody pałają do siebie nienawiścią. Jak do tej pory jednak żadna konfrontacja nie była możliwa. Obie wyspy dzieli kanał zamieszkany przez potwora o imieniu Goorza. Nikt nawet nie odważył się podejść do brzegu i długo jeszcze nic by się nie wydarzyło, gdyby nie wynaleziono tajemniczych latających maszyn. Teraz Ty musisz zapewnić Eolii bezpieczeństwo i poprowadzić jej synów do zwycięstwa. Zaraz po niekoniecznie genialnym intrze musimy podjąć pierwszą decyzję. Z lewej strony ekranu mamy do wyboru kilka scenariuszy:

0. The realm of Eolia. Jest to podstawowy scenariusz gry. Wraz z oponentem (komputerem) startujemy od samego początku. W tej grze szanse są bardzo wyrównane.

1. The golden age. Mamy rok 7272. Eolia za najlepszych czasów. Niezwykle sprzyjający klimat, pełny skarbiec, właściwe rządy. Sharkaanię, odwiecznego wroga trawi głód.

2. The decline. Jest już rok 7406. Życie okazało się za proste w Eolii podczas złotego wieku. Twoi poddani stopniowo zapomnieli o tradycyjnych wartościach, popadli w lenistwo, wciąż domagają się tylko chleba i zabawy.

3. The great war. Rok 7485. Twoi wrogowie z Sharkaanii pragną "ostatecznie rozwiązać kwestię Eolii". Wojskowi narzucają swoją wolę w parlamencie. Kraj pogrążony jest w wojnie, a jego młodzież opuszcza swe domy by walczyć i zginąć.

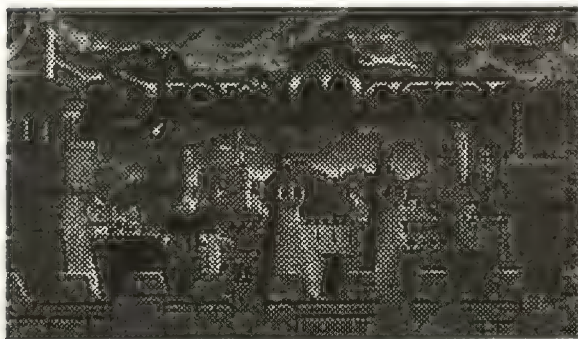
4. The great famine. Początek 7617. Bolesny czas dla dzieci Eolii. Głód i straszliwe plagi dziesiątkują ludność. W Sharkaanii ludzie tańczą i świętują twoje nieszczęścia.

5. The end of a World. Ostatnie lata wojny, rok 7782. Eolia jest pogrążona w chaosie. Musisz wykazać się nie lada umiejętnościami by zdążyć uratować swój świat. Sharkaanie wietrząc łatwe zwycięstwo planują inwazję.

Wybierając jeden z tych scenariuszy możesz utrudnić lub uprościć swoje zadanie, zyskując lub tracąc z wyboru przewagę nad przeciwnikiem. Oprócz tego z prawej strony ekranu znajduje się dodatkowa pięciostopniowa skala trudności oraz możliwość rezygnacji z animowanych sekwencji arcade. Ale oto zasiadasz już w swoim fotelu przewodniczącego, a wokół ciebie twoich siedmiu ministrów. Każdy z tych panów zajmuje się osobną dziedziną. Poczynając od lewej zasiadają: Commander (zajmie sie obronnością), Inquisitor (szef tajnej policji), Leonaardo (naczelny konstruktor i wynalazca), Joker (facet od rozrywki i kultury), High Constable (ekonomista, handlowiec), Master Millar (myśli głównie o tym jak wyżywić Eolię), Ecclesiast (gruba ryba, czarodziej i pierwszy kapłan zarazem). Na sali obecny jest także twój osobisty doradca (zawsze posiada ścisłe dane dotyczące państwa) oraz Skryba gotowy zapisać grę na dysk. Po uaktywnieniu każdej z tych osób ukaże się nam kilka ikonek oznaczających czynności możliwe do wykonania za ich pomocą. I tak Commander pomoże nam wybudować kilka strategicznych obiektów, rozmieścić na lotniskach żołnierzy oraz latające maszyny, dokonać zaciągu no i oczywiście poprowadzić wojnę. Korzystając z opcji Inquisitor'a mamy do dyspozycji całą gamę głównie szpiegowskich usług. Tutaj możemy zaopatrzyć się w aktualne mapy Sharkaani, załatwić inne ciemne sprawki, jak kaptowanie zbiegów z wrogiej wyspy czy pertraktacje z płatnym mordercą. Można też za pomocą gołębiej poczty dowiedzieć się co dzieje się w naszych miastach lub sercach naszych ministrów. Od dobrej roboty Master Millar'a mocno zależą nastroje Eolian. To on zakłada farmy, decyduje o hodowli zbóż, bydła, miodu, wznosi młyny i dotuje niektóre inicjatywy. Nadwyżki produktów jego pracy można sprzedać na giełdzie. Zajmie się tym High Constable. Handel, obrót wszelkiego rodzaju dobrami, rynek to jego żywioły. Na giełdzie można kupić i sprzedać prawie wszystko: części do budowy latających machin, żywność. Sympatyczny gość w centrum ekranu potwierdzi młotkiem każdą transakcję. Można też ubić okazyjny interes z handlowcami z innych miast, zliczyć dochody z miast Eolii. Pamiętaj jednak, że jak na każdej giełdzie ceny nie są tutaj stałe. W zależności od koniunktury rosną i opadają. Można więc stracić, ale przy odrobinie wprawy zarobić krocie. Leonaardo od pierwszego wejrzenia kojarzy nam się ze sławnym mistrzem da Vinci i nie bez powodu. Po wejściu do jego warsztatu zauważysz na pewno dwóch uczniów. Tak się złożyło, że podzielili się oni dosyć specyficznie pracą: jeden niszczy, a drugi buduje. Po lewej stronie ekranu mistrz może zadecydować ilu i w jakim mieście przyjąć uczniów do szkół. Na środku znajduje się wykaz aktualnie osiągalnych konstrukcji. Sam Leonaardo zakończył właśnie cztery najnowsze projekty. Pozostało jeszcze tylko wykończenie konstrukcji drobnymi ale niezwykle ważnymi szczegółami. Po wybraniu jednego z modeli musisz uzbroić go w takie rzeczy jak silnik, śmigła, skrzydła, żagle, balony, tarcze

64





Strona tytułowa...

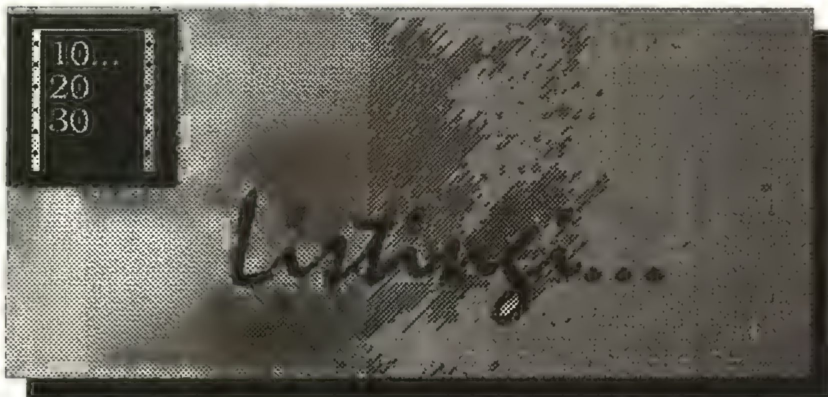
zapyla uprawy, a co najważniejsze przenosi twoje latające machiny i armie. Ecclesiast potrafi również wzniecić tajfuny i błyskawice pomocne zwłaszcza w zwalczaniu wroga na twoim terytorium. Oprócz rozrywek Joker'a, odprawiane przez Ecclesiast'a modły w świątyniach, znacznie poprawiają humory gawiedzi. Oprócz powszechnej szczęśliwości i dobrobytu na Eolii, twoim celem jest także walka z twoimi odwiecznymi wrogami. W tym celu przede wszystkim musisz wybudować armadę swoich statków powietrznych. Jeżeli zdobyłeś już potrzebne do budowy materiały oraz zrekrutowałeś odpowiednią ilość załogi, udaj się do Leonaard'a i zleć mu wykonanie potrzebnych ci machin. Następnie umieść gotowe do walki odziały na przygotowane wcześniej lotniska. Teraz możesz już startować by zmierzyć się z wrogiem. Taktyka walki zależy oczywiście od ciebie. Możesz zbombardować miasta Sharkaani, wysadzić silny desant, złupić cały kraj. Osobiście podobały mi się sekwencje walki powietrznej. Oprócz sporej katapulty, dysponujesz dwoma niezwykle orginalnymi "karabinami maszynowymi". Statek najwygodniej sterować jest za pomocą jojstick'a. Całość stanowi miłą odskocznię od problemów dnia codziennego Eolii. I oby było na tyle jeżeli chodzi o naszą legendę. Oceniając gry tego rodzaju wychodzą z tego przeważnie dwa opisy osobne dla arcade i strategy. Domieszka sekwencji zręcznościowych jest jednak nie wiadomo dlaczego bardzo mała. Ciężko więc w przypadku arcade napisać coś więcej niż to, że część ta jest zrobiona bardzo starannie, z ładną grafiką i przyjemną animacją. Jeżeli chodzi o część strategy to również ciężko się tu dopatrzeć rażących uchybień. Kilka ciekawych pomysłów (gołębia poczta, licytacja na giełdzie) ubranych w bardzo przyzwoitą grafikę zostawia naprawdę miłe wrażenie. Każdemu z miejsc które odwiedzamy towarzyszą specyficzne odgłosy gwaru, ptasiego krzyku, stuku narzędzi. Wszystko znajduje swoją pełnię w świątyni w czasie modłów odprawianych przez Ecclesiast'a. Praktycznie każda postać po kliknięciu myszki aktywnie włącza się do ceremoni. Mamy tam do dyspozycji chóry, organy, gong i wiele innych. Od inwencji gracza zależy więc cała uroczystość. Stosunkowo oryginalny wydaje się być scenariusz całej gry ze wszystkimi elementami fantasy. Pewny zawód może spotkać jednak graczy z małym doświadczeniem w dziedzinie tego rodzaju gier strategicznych. Nawet na najłatwiejszym poziomie łatwo podzielić los poprzedniego przewodniczącego i wykrwawić się na śmierć (zejście naturalne wykluczone). Cóż, lud potrafi czasami spontanicznie wyrażać swoje uczucia. W sumie Storm Master jawi się jako coś co warto mieć w pudełku z dyskami. Tym bardziej, że gra zajmuje tylko jedną dyskietkę. Ktoś kiedyś pomyślał i doszedł do wniosku, że nawet najlepsza gra strategiczna może niekiedy znużyć. Uzupełnił więc ją o elementy arcade, a patent ten w wydaniu Storm Mastera może kosztować was niejedną spędzoną przy komputerze godzinę.

itd. Tak jak i dziś powinieneś mieć w swojej armadzie statki bombowce, transportery, myśliwce, wielo-zadaniowe. Musisz dokonać wyboru między prędkością (ciężki silnik i śmigła), a udźwigiem okrętu (ilością zabranej na pokład załogi). Kompromisem będzie więc także montaż ciężkich tarcz. Do tak zaprojektowanej konstrukcji możemy teraz wprowadzić załogę. Z wcześniej zrekrutowanych przez Commander'a ludzi tworzymy nasz przyszły zespół. Dobieramy więc pośród: captain (dowodzi pojedynczym okrętem), pilot (lata na wszystkim czym latać się da), rifleman (obsługuje "karabin maszynowy"), catapultor (spec od każdej katapulty), soldier (żołnierze przydają się do walk naziemnych), cook (najważniejsza postać, od niego zależy dobre samopoczucie naszych ludzi). Po skompletowaniu załogi możemy przystąpić do testów. Jeżeli załoga i konstrukcja są w porządku, poszybujesz wysoko ponad głowami obserwatorów. W przeciwnym razie maszyna niechybnie runie w dół. W czasie testów cały czas otrzymujesz dane o szacunkowej prędkości, sile ognia, opancerzeniu, wadze całkowitej, udźwigu i stanie załogi. W przypadku pomyślnego zakończenia testów dana konstrukcja otrzymuje atest. Teraz można już przystąpić do jej produkcji. Żadno jednak z tych działań nie powiodłoby się bez życzliwości bogów. O to dba w Eolii Ecclesiast. Od niego zależy naprawdę bardzo wiele. Przede wszystkim nie na darmo nosi on miano Storm Master'a. Jeżeli postawisz mu godną świątynię pomoże ci przywołać błogosławione wiatry, odprawić modły o żyzną ziemię. Pamiętaj, że wiatr niesie Eolii życie. To on obraca łopaty twoich młynów,

McGreg

Tu podejmowane są istotne decyzje...





Wśród użytkowników Amig nawet nieźle znających assembler, krążą pogłoski o rzekomo trudnym wykorzystaniu procedur zawartych w bibliotekach systemu operacyjnego. Aby zdementować te niesłuszne zarzuty, postaram się, aby w Kebabie znalazły się w dziale "Listingi" programy w postaci tekstu źródłowego do samodzielnego "wklepania", wykorzystujące wspomniane biblioteki. Programy będą tak dobrane, aby zarówno były interesujące dla przeciętnego użytkownika, jak i wytrawnego programisty. Dzisiaj prezentuję program do odczytu zawartości dyskietki. Przedstawiony wydruk należy uważnie przepisać, a następnie zasemblować (komenda A) na "Seka Assembler" lub po niewielkich przeróbkach na każdym innym. Następnie komendą WO zapisać na dyskietce pod nazwą "Dir". Po uruchomieniu zapisanego już programu, zostaną wypisane programy znajdujące się na "DF0:". Oczywiście istnieje możliwość odczytu każdego innego katalogu. W tym celu należy przy uruchomianiu za nazwą "Dir" podać nazwę żądanego katalogu.

UWAGA!! Program po zasemblowaniu ma długość 1104 ($0450) bajtów. Długość ta zostaje wypisana po wykonaniu komendy WO.

Krzysztof Kobus

Listing 1

```
***************************
*    Odczyt katalogu*
*    *
* written by K.K./Quartet  *
*    (c) 1992 Kebab       *
***************************

;Tekst zrodlowy dla SEKA
Assembler

;> Exec.library
OldOpenLibrary:=-408
CloseLibrary:   =-414

;> Dos.library
Write:      =-48
Output:=-60
Lock: =-84
UnLock:=-90
Examine:=-102
ExNext:=-108

ACCESS_READ=-2
;------------

Start:
     cmp.l #1,d0
     beq  NoName

     lea   DirName(pc),a1
NextLetter:
     move.b  (a0)+,d1
     cmp.b #$0a,d1
     beq  EndIt
     move.b  d1,(a1)+
     bra   NextLetter
EndIt:
     clr.b (a1)+

NoName:
     move.l  4.w,a6
     lea   DosName(pc),a1
     jsr   OldOpenLibrary(a6)
     move.l  d0,DosBase
     tst.l d0
     beq  Error

     move.l  DosBase(pc),a6
     jsr   Output(a6)
     tst.l d0
     beq  Error
     move.l  d0,TheHandle

     move.l  #DirName,d1
     moveq
#ACCESS_READ,d2
     jsr   Lock(a6)
     move.l  d0,TheLock
     tst.l d0
     beq  Nieznany

     move.l  DosBase(pc),a6
     move.l  d0,d1
     move.l  #FileInfo,d2
     jsr   Examine(a6)
     tst.l d0
     beq  Pusty

     tst.l FileInfo+4
     bmi  NieKatalog

NextPosition:
     move.l  DosBase(pc),a6
     move.l  TheLock(pc),d1
     move.l  #FileInfo,d2
     jsr   ExNext(a6)
     tst.l d0
     beq  Pusty

     lea  FileInfo+8(pc),a0
     lea  Bufor(pc),a1
     moveq     #0,d3
Loop:
     addq.b  #1,d3
     move.b  (a0)+,(a1)+
     bne  Loop
     move.b  #$0a,-(a1)

     tst.l FileInfo+4
     bmi  DoDruk

     move.b  #' ',(a1)+
     move.b  #'(',(a1)+
     move.b  #'d',(a1)+
     move.b  #'i',(a1)+
     move.b  #'r',(a1)+
     move.b  #')',(a1)+
     move.b  #$0a,(a1)+
     addq.b  #6,d3

DoDruk:
     move.l  TheHandle(pc),d1
     move.l  #Bufor,d2
     jsr   Write(a6)

     bra  NextPosition

Pusty:
     move.l  DosBase(pc),a6
     move.l  TheLock(pc),d1
     jsr   UnLock(a6)

End:
     move.l  4.w,a6
     move.l  DosBase(pc),a1
     jsr   CloseLibrary(a6)
     moveq     #0,d0
     rts  ;Ostateczny powrot

Error:
     moveq     #0,d0
     rts

Nieznany:
     move.l  TheHandle(pc),d1
     move.l  #Note1,d2
     move.l  #20,d3
     jsr   Write(a6)
     bra  End

NieKatalog:
     move.l  TheHandle(pc),d1
     move.l  #Note2,d2
     move.l  #23,d3
     jsr   Write(a6)
     bra  Pusty
;------------
align4

FileInfo:  blk.b 260,0
Bufor:     blk.b 108,0
TheLock:   dc.l  0
TheHandle: dc.l  0
DosBase:   dc.l  0
DosName:   dc.b  'dos.library',0
Note1:     dc.b  'Katalog
nieznany!!!',$0a
Note2:     dc.b  'To nie jest
katalog!!!',$0a
DirName:   dc.b  'df0:',0
     blk.b 250,0
```



Listing 2

"1AR-TOOL SAVER " $0801-$0B1F

```
:0801 0B 08 0A 00 9E 32 30 35 (7C)
:0809 39 00 A0 00 C8 98 A2 08 (50)
:0811 20 BA FF A9 08 A2 17 A0 (E3)
:0819 0B 20 BD FF 20 C0 FF A2 (C8)
:0821 01 20 C9 FF A9 4E 85 FB (5D)
:0829 A9 08 85 FC A0 00 B1 FB (38)
:0831 20 D2 FF E6 FB D0 02 E6 (97)
:0839 FC A5 FC C9 0B 90 ED A5 (D9)
:0841 FB C9 17 90 E7 A9 01 20 (DB)
:0849 C3 FF 4C CC FF B8 02 20 (7F)
:0851 13 EE 8D 40 03 EE BC 02 (C6)
:0859 D0 03 EE BD 02 EE 20 D0 (F3)
:0861 CE 20 D0 A5 90 F0 E8 A9 (8B)
:0869 01 2C C3 FF 20 28 F5 A9 (9A)
:0871 37 85 01 A9 1B 8D 11 D0 (2D)
:0879 4C 40 03 D0 4F 4C 4F 4E (82)
:0881 55 53 00 40 A9 F6 8D 29 (B8)
:0889 03 A0 00 84 90 8C 20 D0 (5C)
:0891 8C 11 D0 F0 BA 00 00 8B (71)
:0899 E3 25 80 7C A5 1A A7 E4 (C4)
:08A1 A7 86 AE 00 00 00 00 4C (C6)
:08A9 48 B2 00 31 EA EC 02 EC (A9)
:08B1 02 4A F3 91 F2 0E F2 50 (98)
:08B9 F2 33 F3 57 F1 CA F1 ED (BE)
:08C1 02 78 20 53 E4 A2 1F BD (68)
:08C9 30 FD 9D 14 03 CA 10 F7 (15)
:08D1 58 A9 00 85 9D 8D 20 D0 (56)
:08D9 8D 21 D0 AA A9 01 9D 00 (66)
:08E1 DA 9D 00 D8 9D 00 D9 9D (45)
:08E9 00 DB 8A 9D 00 06 E8 D0 (B5)
:08F1 EB A2 9F A9 20 9D 48 07 (27)
:08F9 CA E0 FF D0 F6 4C EB 03 (E3)
:0901 17 93 CB C5 C2 C1 C2 20 (5A)
:0909 C6 4F 4E 54 2D C3 41 54 (8A)
:0911 43 48 45 52 20 28 C1 43 (F3)
:0919 54 49 4F 4E 2D D2 45 50 (5D)
:0921 4C 41 59 20 54 4F 4F 4C (8A)
:0929 29 00 28 C6 31 2F C6 33 (FC)
:0931 29 24 44 30 31 38 2C 24 (D0)
:0939 31 38 20 20 28 C6 35 29 (EA)
:0941 24 44 44 30 30 2C 24 30 (F6)
:0949 33 20 2F D2 C5 D4 D5 D2 (CE)
:0951 CE 00 C5 4E 54 45 52 20 (2F)
:0959 C6 49 4C 45 4E 41 4D 45 (01)
:0961 3A 00 C9 60 90 03 29 7F (78)
:0969 60 29 3F 60 A2 00 BD 81 (BE)
:0971 03 F0 09 20 E1 03 9D 98 (7A)
:0979 07 E8 D0 F2 A2 00 BD A9 (2E)
:0981 03 F0 09 20 E1 03 9D C0 (CA)
:0989 07 E8 D0 F2 78 A9 1A 8D (0D)
:0991 14 03 A9 04 8D 15 03 58 (D3)
:0999 4C 68 04 A9 D8 CD 12 D0 (72)
:09A1 D0 FB A9 16 8D 18 D0 A9 (0C)
:09A9 93 8D 00 DD A9 E0 CD 12 (8B)
:09B1 D0 D0 FB A2 0A CA D0 FD (29)
:09B9 EA CE 20 D0 CE 21 D0 A2 (74)
:09C1 0A CA D0 FD EE 21 D0 EE (58)
:09C9 20 D0 2C 12 D0 30 FB AD (D3)
:09D1 7F 03 8D 18 D0 AD 80 03 (1C)
:09D9 8D 00 DD 4C 31 EA C9 0A (76)
:09E1 B0 03 09 30 60 38 E9 09 (52)
:09E9 60 AD 7F 03 48 4A 4A 4A (AF)
:09F1 4A 20 5D 04 8D CE 07 68 (B1)
:09F9 29 0F 20 5D 04 8D CF 07 (60)
:0A01 AD 80 03 29 0F 20 5D 04 (1B)
:0A09 8D DE 07 20 E4 FF F0 FB (C7)
:0A11 C9 85 D0 0C AD 7F 03 18 (BE)
:0A19 69 10 8D 7F 03 4C 68 04 (1E)
:0A21 C9 86 D0 18 AD 7F 03 48 (80)
:0A29 29 F0 8D 7F 03 68 18 69 (4E)
:0A31 02 29 0F 0D 7F 03 8D 7F (50)
:0A39 03 4C 68 04 C9 87 D0 10 (6D)
:0A41 AD 80 03 18 69 01 29 03 (AB)
:0A49 09 90 8D 80 03 4C 68 04 (F2)
:0A51 C9 0D D0 B7 A2 00 BD D1 (67)
:0A59 03 F0 09 20 E1 03 9D 70 (23)
:0A61 07 E8 D0 F2 A2 16 A9 13 (5F)
:0A69 20 D2 FF A9 11 20 D2 FF (A3)
:0A71 CA D0 FA A9 0F 85 D3 A2 (B5)
:0A79 00 20 CF FF C9 0D F0 08 (37)
:0A81 9D 00 02 E8 E0 10 90 F1 (06)
:0A89 86 02 78 A9 31 8D 14 03 (10)
:0A91 A9 EA 8D 15 03 A9 16 8D (1A)
:0A99 18 D0 A9 93 8D 00 DD 58 (2E)
:0AA1 AD 80 03 0A 0A 0A 0A 0A (8D)
:0AA9 0A 85 FC A9 00 85 FB AD (C2)
:0AB1 7F 03 29 0E 0A 0A 05 FC (64)
:0AB9 85 FC A9 01 A8 A2 08 20 (8B)
:0AC1 BA FF A5 02 A2 00 A0 02 (14)
:0AC9 20 BD FF 20 C0 FF A2 01 (1A)
:0AD1 20 C9 FF A9 00 20 D2 FF (A4)
:0AD9 A9 20 20 D2 FF A9 37 85 (0E)
:0AE1 01 A2 08 A0 00 78 E6 01 (EA)
:0AE9 B1 FB C6 01 20 D2 FF E6 (A5)
:0AF1 FB D0 F0 E6 FC CA D0 EB (AE)
:0AF9 58 A9 01 20 C3 FF 20 CC (39)
:0B01 FF 4C 70 03 20 43 41 54 (98)
:0B09 48 45 52 20 42 59 20 50 (1C)
:0B11 4F 4C 4F 4E 55 53 31 41 (22)
:0B19 52 2D 54 4F 4F 4C BD BD (6E)
```

Listing 3

"2AR-TOOL SAVER " $0801-$0C09

```
:0801 0B 08 0A 00 9E 32 30 35 (7C)
:0809 39 00 A0 00 C8 98 A2 08 (50)
:0811 20 BA FF A9 08 A2 01 A0 (49)
:0819 0C 20 BD FF 20 C0 FF A2 (C9)
:0821 01 20 C9 FF A9 4E 85 FB (5D)
:0829 A9 08 85 FC A0 00 B1 FB (38)
:0831 20 D2 FF E6 FB D0 02 E6 (97)
:0839 FC A5 FC C9 0C 90 ED A5 (DE)
:0841 FB C9 01 90 E7 A9 01 20 (99)
:0849 C3 FF 4C CC FF B8 02 20 (7F)
:0851 13 EE 8D 40 03 EE BC 02 (C6)
:0859 D0 03 EE BD 02 EE 20 D0 (F3)
:0861 CE 20 D0 A5 90 F0 E8 A9 (8B)
:0869 01 2C C3 FF 20 28 F5 A9 (9A)
:0871 37 85 01 A9 1B 8D 11 D0 (2D)
:0879 4C 40 03 D0 4F 4C 4F 4E (82)
:0881 55 53 00 40 A9 F6 8D 29 (B8)
:0889 03 A0 00 84 90 8C 20 D0 (5C)
:0891 8C 11 D0 F0 BA 00 00 8B (71)
:0899 E3 25 80 7C A5 1A A7 E4 (C4)
:08A1 A7 86 AE 00 00 00 00 4C (C6)
:08A9 48 B2 00 31 EA EC 02 EC (A9)
:08B1 02 4A F3 91 F2 0E F2 50 (98)
:08B9 F2 33 F3 57 F1 CA F1 ED (BE)
:08C1 02 78 20 53 E4 A2 1F BD (68)
:08C9 30 FD 9D 14 03 CA 10 F7 (15)
:08D1 58 A2 3F A9 08 9D 00 D4 (4C)
:08D9 A9 00 9D 00 D4 CA 10 F3 (49)
:08E1 E8 9D 00 D8 9D 00 D9 9D (53)
:08E9 00 DA 9D 00 DB E8 D0 F1 (6B)
:08F1 8D 20 D0 8D 21 D0 85 9D (7A)
:08F9 A9 17 8D 18 D0 A9 80 0D (CD)
:0901 91 02 8D 91 02 4C 5E 04 (0E)
:0909 C9 60 B0 03 29 3F 60 29 (E6)
:0911 7F 60 AE 5D 04 BC 53 04 (B8)
:0919 B9 C0 07 49 80 99 C0 07 (22)
:0921 60 CB C5 C2 C1 C2 20 C4 (C8)
:0929 49 47 49 2D C3 41 54 43 (51)
:0931 48 45 52 20 28 C1 43 54 (45)
:0939 49 4F 4E 20 D2 45 50 4C (DB)
:0941 41 59 20 54 4F 4F 4C 29 (AE)
:0949 00 C3 D2 D3 D2 2F D2 C5 (B4)
:0951 D4 D5 D2 CE 3B 20 D3 2D (9A)
:0959 53 41 56 45 20 20 28 57 (7D)
:0961 29 42 59 20 D0 4F 4C 4F (18)
:0969 4E 55 53 00 D0 4C 41 59 (CA)
:0971 49 4E 47 20 53 41 4D 50 (74)
:0979 4C 45 21 20 D2 D5 CE 2F (6D)
:0981 D3 D4 CF D0 20 54 4F 20 (73)
:0989 41 42 4F 52 54 20 00 D3 (88)
:0991 54 41 52 54 3A 24 30 38 (C0)
:0999 30 33 3B 20 C5 4E 44 3A (C2)
:09A1 24 46 46 30 30 3B 20 D3 (B6)
:09A9 50 45 45 44 3A 24 37 30 (66)
:09B1 00 C5 4E 54 45 52 20 C6 (D3)
:09B9 49 4C 45 4E 41 4D 45 3A (70)
:09C1 20 20 20 20 20 20 20 20 (4A)
:09C9 20 20 20 20 20 20 20 20 (52)
:09D1 20 20 20 00 07 08 09 0A (7C)
:09D9 12 13 14 15 1F 20 00 A2 (15)
:09E1 27 A9 01 9D 70 DB 9D C0 (77)
:09E9 DB 9D 48 DB A9 20 9D 70 (23)
:09F1 07 9D 98 07 9D C0 07 CA (31)
:09F9 10 E7 A2 00 BD A0 03 F0 (CC)
:0A01 09 20 87 03 9D 70 07 E8 (17)
:0A09 D0 F2 A2 00 BD 0E 04 F0 (4E)
:0A11 09 20 87 03 9D C0 07 E8 (07)
:0A19 D0 F2 A2 00 BD C8 03 F0 (B3)
:0A21 09 20 87 03 9D 98 07 E8 (27)
:0A29 D0 F2 A9 27 A9 03 9D 98 (E8)
:0A31 DB CA 10 F8 20 91 03 20 (D5)
:0A39 E4 FF F0 FB C9 1D D0 15 (D4)
:0A41 20 91 03 AD 5D 04 18 69 (23)
:0A49 01 C9 0A 90 02 A9 00 8D (AC)
:0A51 5D 04 4C B3 04 C9 9D D0 (05)
:0A59 10 20 91 03 CE 5D 04 10 (42)
:0A61 05 A9 09 8D 5D 04 4C B3 (A6)
:0A69 04 C9 53 D0 03 4C BD 05 (6C)
:0A71 C9 0D F0 13 20 6D 06 B0 (52)
:0A79 BE AE 5D 04 BC 53 04 29 (C6)
:0A81 3F 99 C0 07 4C C2 04 20 (7C)
:0A89 91 03 A2 00 BD EB 03 F0 (D8)
:0A91 09 20 87 03 9D 98 07 E8 (97)
:0A99 D0 F2 20 1F 05 4C 58 05 (A4)
:0AA1 AC C7 07 AD C8 07 20 51 (28)
:0AA9 06 85 FC AC C9 07 AD CA (89)
:0AB1 07 20 51 06 85 FB AC D2 (CC)
:0AB9 07 AD D3 07 20 51 06 85 (91)
:0AC1 FE AC D4 07 AD D5 07 20 (49)
:0AC9 51 06 85 FD AC DF 07 AD (E2)
:0AD1 E0 07 20 51 06 AA 85 02 (3A)
:0AD9 60 A9 37 85 01 78 A0 01 (8B)
:0AE1 A0 01 8C 0D DD 8C 0E DD (48)
:0AE9 88 8C 05 DD 8C 15 D0 8E (70)
:0AF1 04 DD A0 00 E6 01 B1 FB (CC)
:0AF9 C6 01 48 4A 4A 4A 4A AE (6F)
:0B01 0D DD F0 FB 8D 18 D4 8D (14)
:0B09 20 D0 68 29 0F AE 0D DD (52)
:0B11 F0 FB 8D 18 D4 8D 20 D0 (DB)
:0B19 AD 01 DC 10 12 E6 FB D0 (C2)
:0B21 02 E6 FC A5 FC C5 FE D0 (7E)
:0B29 CB A5 FB C5 FD D0 C5 A9 (CA)
:0B31 00 8D 18 D4 8D 0E DD 8D (76)
:0B39 20 D0 58 4C 99 04 A2 00 (BF)
:0B41 BD 30 04 F0 09 20 87 03 (EB)
:0B49 9D 98 07 E8 D0 F2 A9 13 (C9)
:0B51 20 D2 FF 20 91 03 A2 17 (AA)
:0B59A91120D2FFCAD0FA(0E)
```

```
:0B61 A9 0F 85 D3 A2 00 20 CF (90)
:0B69 FF C9 0D F0 08 9D 00 02 (D2)
:0B71 E8 E0 10 90 F1 8E 07 06 (FE)
:0B79 20 1F 05 A9 01 A8 A2 08 (38)
:0B81 20 BA FF A2 00 A0 02 A9 (BB)
:0B89 20 F0 3E 20 BD FF 20 C0 (59)
:0B91 FF A2 01 20 C9 FF A5 FB (A4)
:0B99 20 D2 FF A5 FC 20 D2 FF (5B)
:0BA1 A0 00 78 E6 01 B1 FB C6 (84)
:0BA9 01 20 D2 FF 8D 20 D0 E6 (C8)
:0BB1 FB D0 02 E6 FC A5 FC C5 (CB)
:0BB9 FE D0 E7 A5 FB C5 FD D0 (9B)
:0BC1 E1 A9 01 20 C3 FF 20 CC (8B)
:0BC9 FF A9 00 D0 20 D0 58 4C (A1)
:0BD1 99 04 20 61 06 85 02 98 (6B)
:0BD9 20 61 06 0A 0A 0A 0A 05 (DC)
:0BE1 02 60 C9 30 B0 04 18 69 (41)
:0BE9 09 60 29 0F 38 60 C9 30 (CB)
:0BF1 90 FA C9 47 B0 F6 C9 41 (B2)
:0BF9 B0 04 C9 3A B0 EE 18 60 (AB)
:0C01 32 41 52 2D 54 4F 4F 4C (72)
```

Listing 4

W związku z faktem błednego zamieszczenia KOREKTORA 64 w pierwszym numerze KEBABA, postanowiliśmy umieścić jego listing w BASIC ponownie, tym razem bez błędów. Za kłopot serdecznie przepraszamy.

```
4 REM ******************************
5 REM ** KOREKTOR KODU
MASZYNOWEGO **
6 REM ** (C)1991 COMMODORE-
KEBAB **
7 REM ******************************
8 :
10 AD=20480:PRINT
CHR$(147);TAB(7);"(21727 - KONIEC)"
20 READ A$:IF A$="GG" THEN 62
30 GOSUB 40:POKE
AD,A:AD=AD+1:PRINT
CHR$(19);AD:K=K+A:GOTO 20
40 B$=LEFT$(A$,1):GOSUB 60:A=C*16
50 B$=RIGHT$(A$,1):GOSUB
60:A=A+C:RETURN
60 IF ASC(B$)>64 THEN C=ASC(B$)-
55:RETURN
61 C=ASC(B$)-48:RETURN
62 IF K<>126480 THEN PRINT"ZLE
DANE!!":STOP
70 PRINT:PRINT"TASMA CZY DYSK
(T/D)":PRINT:PRINT
75 GET A$:IF A$<>"T" AND A$<>"D"
THEN 75
76 IF A$="D" THEN D=8:GOTO 78
77 D=1
78 POKE 20483,D:SYS20480
79 :
1000 DATA 4C,27,51,08,00,01,08,01
1001 DATA 08,0C,0D,0F,10,12,13,15
1002 DATA 16,18,19,1B,1C,1E,1F,21
1003 DATA 22,25,26,AE,04,50,BD,09
1004 DATA 50,AA,BD,90,05,49,80,9D
1005 DATA 90,05,60,A9,00,85,52,A5
1006 DATA 50,20,3C,50,0A,0A,0A,0A
1007 DATA 85,52,A5,51,C9,30,B0,05
1008 DATA 18,69,09,D0,03,38,E9,30
1009 DATA 05,52,85,52,60,48,4A,4A
1010 DATA 4A,4A,20,5A,50,85,50,68
1011 DATA 29,0F,C9,0A,90,05,38,E9
1012 DATA 09,D0,02,09,30,85,51,60
1013 DATA C9,30,B0,02,38,60,C9,47
1014 DATA B0,FA,C9,3A,90,F1,C9,41
1015 DATA 90,F2,18,60,93,8E,08,90
1016 DATA 20,20,20,20,20,20,20,12
1017 DATA 20,2A,2A,20,4B,45,42,41
1018 DATA 42,2D,4B,4F,52,45,4B,54
1019 DATA 4F,52,20,43,36,34,20,2A
1020 DATA 2A,20,92,0D,20,20,20,20
1021 DATA 20,20,28,57,29,31,39,39
1022 DATA 31,20,42,59,20,50,41,57
1023 DATA 45,4C,20,53,4F,4C,54,59
1024 DATA 53,49,4E,53,4B,49,00,45
1025 DATA 44,49,54,3E,4C,4F,43,41
1026 DATA 54,49,4F,4E,3A,24,30,30
1027 DATA 30,30,2D,24,30,30,30,30
1028 DATA 0D,43,54,52,4C,2B,4C,20
1029 DATA 2D,4C,4F,41,44,0D,43,54
1030 DATA 52,4C,2B,53,20,2D,53,41
1031 DATA 56,45,0D,43,54,52,4C,2B
1032 DATA 52,20,2D,52,45,53,54,41
1033 DATA 52,54,00,30,30,20,30,30
1034 DATA 20,30,30,20,30,30,20,30
1035 DATA 30,20,30,30,20,30,30,20
1036 DATA 30,30,20,28,20,20,29,A9
1037 DATA 0C,8D,20,D0,8D,21,D0,A9
1038 DATA 01,8D,86,02,A9,37,85,01
1039 DATA A9,7C,A0,50,20,1E,AB,A2
1040 DATA 80,86,9D,A9,01,9D,50,D8
1041 DATA 9D,10,D9,9D,10,DA,9D,00
1042 DATA DB,E8,D0,F1,A2,27,A9,2D
1043 DATA 9D,50,04,9D,B8,05,CA,10
1044 DATA F5,A2,04,BD,C7,50,29,3F
1045 DATA 9D,90,05,A9,00,9D,90,D9
1046 DATA CA,10,F0,E8,BD,CC,50,29
1047 DATA 3F,9D,E0,05,E8,E0,14,D0
1048 DATA F3,A9,0D,85,D6,A9,E0,A0
1049 DATA 50,20,1E,AB,A9,3A,8D,96
1050 DATA 05,AD,08,50,20,4D,50,8D
1051 DATA 98,05,8D,F1,05,A5,50,8D
1052 DATA 97,05,8D,F0,05,AD,07,50
1053 DATA 20,4D,50,8D,9A,05,8D,F3
1054 DATA 05,A5,50,8D,99,05,8D,F2
1055 DATA 05,AD,06,50,20,4D,50,8D
1056 DATA EB,05,A5,50,8D,EA,05,AD
1057 DATA 05,50,20,4D,50,8D,ED,05
1058 DATA A5,50,8D,EC,05,A2,1B,BD
1059 DATA 0B,51,9D,9C,05,CA,10,F7
1060 DATA A9,00,8D,04,50,20,1B,50
1061 DATA 20,76,52,20,E4,FF,F0,FB
1062 DATA C9,1D,D0,10,AD,04,50,C9
1063 DATA 11,B0,ED,20,1B,50,EE,04
1064 DATA 50,4C,E5,51,C9,9D,D0,0E
1065 DATA AD,04,50,F0,DE,20,1B,50
1066 DATA CE,04,50,4C,E5,51,C9,12
1067 DATA D0,03,4C,7F,54,C9,0D,D0
1068 DATA 32,A5,02,F0,2E,AD,07,50
1069 DATA 85,20,AD,08,50,85,21,78
1070 DATA E6,01,A0,07,B9,80,03,91
1071 DATA 20,88,10,F8,C6,01,58,20
1072 DATA DF,52,AD,07,50,18,69,08
1073 DATA 8D,07,50,90,03,EE,08,50
1074 DATA 4C,8C,51,C9,0C,D0,03,4C
1075 DATA B8,53,C9,13,D0,03,4C,55
1076 DATA 54,20,68,50,B0,93,29,3F
1077 DATA A8,20,1B,50,98,9D,90,05
1078 DATA 20,1B,50,4C,F4,51,A9,00
1079 DATA 85,02,A2,00,BC,09,50,B9
1080 DATA 90,05,29,3F,85,50,B9,91
1081 DATA 05,29,3F,85,51,8A,4A,A8
1082 DATA 20,2B,50,99,80,03,E8,E8
1083 DATA E0,12,D0,E0,A2,00,BD,80
1084 DATA 03,8D,AA,52,8E,A8,52,A0
1085 DATA 07,A9,00,18,65,02,85,02
1086 DATA 88,10,F6,E8,E0,08,D0,E6
1087 DATA AD,07,50,18,65,02,85,02
1088 DATA AD,08,50,18,65,02,85,02
1089 DATA CD,88,03,F0,03,A9,00,2C
1090 DATA A9,01,85,02,A2,03,A5,02
1091 DATA 9D,B4,D9,CA,10,FA,60,A0
1092 DATA 06,B9,A0,04,99,78,04,B9
1093 DATA C8,04,99,A0,04,B9,F0,04
1094 DATA 99,C8,04,B9,18,05,99,F0
1095 DATA 04,B9,40,05,99,18,05,B9
1096 DATA 68,05,99,40,05,B9,90,05
1097 DATA 29,3F,99,68,05,C8,C0,28
1098 DATA D0,CF,60,A0,00,20,3D,54
1099 DATA A9,20,A2,23,9D,E4,05,CA
1100 DATA 10,FA,E8,8E,0F,01,A9,22
1101 DATA 8D,E5,05,8D,F6,05,86,02
1102 DATA A6,02,BD,E6,05,09,80,9D
1103 DATA E6,05,20,E4,FF,F0,FB,C9
1104 DATA 0D,F0,16,C9,1D,F0,45,C9
1105 DATA 9D,F0,54,C9,14,F0,61,29
1106 DATA 3F,A6,02,9D,E6,05,4C,8C
1107 DATA 53,A2,10,BD,E5,05,29,3F
1108 DATA C9,20,D0,0E,CA,D0,F4,8E
1109 DATA 0F,01,8A,A2,10,A0,01,4C
1110 DATA BD,FF,20,67,53,A2,00,BD
1111 DATA E6,05,29,3F,C9,1B,B0,02
1112 DATA 09,40,9D,10,01,E8,EC,0F
1113 DATA 01,90,EC,60,A6,02,E0,0F
1114 DATA B0,A8,BD,E6,05,29,3F,9D
1115 DATA E6,05,E6,02,4C,30,53,A6
1116 DATA 02,F0,97,BD,E6,05,29,3F
1117 DATA 9D,E6,05,C6,02,4C,30,53
1118 DATA A6,02,F0,86,A9,20,D0,F0
1119 DATA 20,13,53,20,44,E5,AE,03
1120 DATA 50,E0,08,F0,28,A9,01,A8
1121 DATA 20,BA,FF,A9,00,20,D5,FF
1122 DATA B0,18,AD,3C,03,8D,05,50
1123 DATA AD,3D,03,8D,06,50,AD,3E
1124 DATA 03,8D,07,50,AD,3F,03,8D
1125 DATA 08,50,4C,27,51,AD,0F,01
1126 DATA F0,F8,A0,00,A9,01,20,BA
1127 DATA FF,20,C0,FF,A2,01,20,C6
1128 DATA FF,20,CF,FF,85,AE,A5,90
1129 DATA D0,28,20,CF,FF,85,AF,8D
1130 DATA 06,50,A5,AE,8D,05,50,20
1131 DATA CF,FF,A0,00,91,AE,E6,AE
1132 DATA D0,02,E6,AF,A5,90,F0,EF
1133 DATA A5,AE,8D,07,50,A5,AF,8D
1134 DATA 08,50,A9,01,20,C3,FF,20
1135 DATA CC,FF,4C,27,51,D0,11,A2
1136 DATA 09,BD,E0,50,29,3F,99,E0
1137 DATA 05,E8,C8,C0,04,D0,F2,60
1138 DATA A2,16,88,F0,EC,A0,01,20
1139 DATA 15,53,AE,03,50,A9,01,A8
1140 DATA 20,BA,FF,AD,05,50,85,FB
1141 DATA AD,06,50,85,FC,A9,FB,AE
1142 DATA 07,50,AC,08,50,C6,01,20
1143 DATA D8,FF,E6,01,4C,27,51,A9
1144 DATA 00,85,02,A6,02,BD,EA,05
1145 DATA 09,80,9D,EA,05,20,E4,FF
1146 DATA F0,FB,C9,0D,F0,13,20,68
1147 DATA 50,B0,F2,29,3F,A6,02,9D
1148 DATA EA,05,E8,8A,29,03,4C,81
1149 DATA 54,A6,02,BD,EA,05,29,3F
1150 DATA 9D,EA,05,AD,EA,05,85,50
1151 DATA AD,EB,05,85,51,20,2B,50
1152 DATA 8D,06,50,8D,08,50,AD,EC
1153 DATA 05,85,50,AD,ED,05,85,51
1154 DATA 20,2B,50,8D,05,50,8D,07
1155 DATA 50,4C,27,51,BD,BD,BD,GG
```

Listing 5

Tekst zrodlowy programu Zegar Cyfrowy dla Commodore 64, przeznaczony dla TurboAssembler'a.

```
;Digital Clock 64
;(w) Pawel "Polonus" Soltysinski
1992

        *= $CB20      ;adres programu

EKRAN= $041B
ATRYBUT  = EKRAN+$D400
SEK10 = $DC08
SEKUNDY  = $DC09
MINUTY     = $DC0A
GODZINY  = $DC0B
KOLOR= $01 ;kolor bialy
ZAP1  = $03E0
ZAP2  = ZAP1+13

START LDA #<TXT1
      LDY #>TXT1
      JSR $AB1E  ;drukuj pytanie
      LDX #$00
POBIERZ  JSR $FFCF     ;wprowadz
tekst
      CMP #$0D    ;RETURN ?
      BEQ OK1     ;Tak - idz do OK1
      STA DATA,X  ;Nie - wstaw
znak
      INX
      CPX #$05    ;5 znakow ?
      BCC POBIERZ ;Nie - nastepny
OK1   CPX #$05    ;wprowadzono 5 ?
      BNE START  ;Nie - jeszcze
raz
      LDA #$00    ;ustawienie na
AM
      STA PMFLAG
      LDX #$04
OK2   CPX #$02
      BEQ OK3
      LDA DATA,X
      JSR TEST    ;czy to cyfry?
      BCS START   ;Blad -jeszcze
raz
      AND #$0F
      STA DATA,X
OK3   DEX
      BPL OK2
      CMP #$01    ;procedura
zamiany
      BCC OK4     ;zapisu 24-o godz.
      CMP #$02    ;na 12-to
godzinny
      BEQ LOOP
      LDA DATA+1
      CMP #$03
      BCC OK4
LOOP  DEC DATA
      LDA DATA+1
      SED
      SEC
      SBC #$02
      CLD
      BCS LOOP1
      DEC DATA
LOOP1 AND #$0F
      STA DATA+1
      LDA #$80
      STA PMFLAG   ;ustaw PM
OK4   LDA DATA
      ASL A
      ASL A
      ASL A
      ASL A
      ORA PMFLAG
      ORA DATA+1
      STA GODZINY ;wstaw godzine
      LDA DATA+3
      ASL A
      ASL A
      ASL A
      ASL A
      ORA DATA+4
      STA MINUTY  ;wstaw minuty
      LDA #$00
      STA SEKUNDY ;sekundy na
zero
      STA SEK10   ;start zegara
INIT  SEI   ;procedura usta-
      LDA #$7F     ;wienia przerwan
      STA $DC0D
      LDA #$80     ;$80/$00 -50/60 Hz
      STA $DC0E
      LDA #$01
      STA $D01A
      LDY #$1B
      STY $D011
      LDA #$22     ;IRQ w linii $22
      STA $D012
      LDA #<IRQ    ;ustaw. wektora
      LDY #>IRQ    ;IRQ
      STA $0314
      STY $0315
      LDA #<NMI    ;ustaw. wektora
      LDY #>NMI    ;NMI (r/s-restore)
      STA $0318
      STY $0319
      CLI
      RTS   ;powrot

IRQ   LDX #$0C     ;procedura
zacho-
      LDA EKRAN,X ;wujaca to co
"pod"
      STA ZAP1,X  ;zegarem
      LDA ATRYBUT,X
      STA ZAP2,X
      LDA #KOLOR  ;wskazanie w
zada-
      STA ATRYBUT,X ;nym kolorze
      DEX
      BPL IRQ+2
      LDA GODZINY ;pobranie
godzin
      STA $02
      AND #$70     ;oddzielenie od
PM
      JSR SEK1     ;na kod ekranowy
      STA EKRAN+1 ;i na ekran!
      LDA $02
      JSR SEK2     ;to samo z druga
      STA EKRAN+2 ;cyfra godzin
      LDA MINUTY
      PHA
      JSR SEK1
      STA EKRAN+4 ;podobnie z
      PLA   ;minutami
      JSR SEK2
      STA EKRAN+5
      LDA #$A0     ;spacja+inverse
      STA EKRAN  ;gdzie trzeba...
      STA EKRAN+9
      STA EKRAN+12
      LDA #$2E+$80 ;kropka przed
      STA EKRAN+6  ;sekundami
      LDA $02      ;wyswietlenie
      BMI OK5      ;AM lub PM
      LDA #"A"
      BNE OK6
OK5   LDA #"P"
OK6   AND #$3F
      ORA #$80
      STA EKRAN+10
      LDA #"M"
      AND #$3F
      ORA #$80
      STA EKRAN+11
      LDA SEKUNDY  ;wyswietlenie
      PHA   ;sekund
      JSR SEK1
      STA EKRAN+7
      PLA
      JSR SEK2
      STA EKRAN+8
      LDA SEK10    ;a dwukropek
      CMP #$05     ;ma nam mrugac
      BCS OK7      ;co pol sekundy
      LDA #$3A
      BNE OK8
OK7   LDA #$20
OK8   ORA #$80
      STA EKRAN+3
      LDA #$3C     ;odczekujemy az
OK9   CMP $D012 ;raster minie
      BNE OK9      ;obszar
wyswietla-
      LDX #$0C     ;nia i
odtwarzamy
OK10  LDA ZAP1,X ;dawna zawartosc
      STA EKRAN,X  ;ekranu w tym
      LDA ZAP2,X ;miejscu
      STA ATRYBUT,X
      DEX
      BPL OK10
OK11  INC $D019
      JMP $EA31

TEST  CMP #"0"     ;podprogram
      BCC TEST1    ;"testuj, czy
      CMP #$3A     ;cyfra"
      BCC TEST1+1
TEST1 SEC    ;C=1 - blad
      RTS

SEK1  LSR A;obrobka przed
      LSR A;umieszczeniem
      LSR A;znaku na ekranie
      LSR A
SEK2  AND #$0F
      ORA #$30+$80 ;cyfra+inverse
      RTS

NMI   LDA #$37     ;obsluga NMI
      STA $01      ;r/s-restore
      JSR $FDA3
      JSR $FD15
      JSR $E518
      JSR INIT     ;wlaczyc wyswie-
      LDX #$FB     ;tlanie ponownie
      TXS
      JMP $A474   ;hop do BASIC!

TXT1  .BYTE $0D,$05
      .TEXT "PODAJ GODZINE
(NP.09:13"
      .TEXT "?"
      .BYTE 0

PMFLAG      .BYTE 0      ;wskaznik
dla PM/AM
DATA ;od tego miejsca
      ;beda dane z kla-
      ;wiatury
```

Listing 6

```
"TIMER C64"      $CB20-$CCC2

0 REM *
1 REM * ZEGAR CYFROWY DLA
C64
2 REM * (W) Pawel Soltysinski
3 REM *
4 :
10 FOR T=52000 TO 52417:READ A$
20 GOSUB 50:POKE T,A:B=B+A:NEXT
T
30 IF B<>46269 THEN PRINT "ZLE
DANE!":STOP
40 SYS 52000:REM START
PROGRAMU
50 B$=LEFT$(A$,1):GOSUB 70:A=C*16
60 B$=RIGHT$(A$,1):GOSUB
70:A=A+C:RETURN
70 C=ASC(B$):IF C>64 THEN C=C-
55:RETURN
80 C=C-48:RETURN
90 :
1000 DATA
A9,A5,A0,CC,20,1E,AB,A2,00,20
1001 DATA
CF,FF,C9,0D,F0,08,9D,C2,CC,E8
1002 DATA
E0,05,90,F1,E0,05,D0,E4,A9,00
1003 DATA
8D,C1,CC,A2,04,E0,02,F0,0D,BD
1004 DATA
C2,CC,20,7C,CC,B0,D1,29,0F,9D
1005 DATA
C2,CC,CA,10,EC,C9,01,90,25,C9
1006 DATA
02,F0,07,AD,C3,CC,C9,03,90,1A
1007 DATA
CE,C2,CC,AD,C3,CC,F8,38,E9,02
1008 DATA
D8,B0,03,CE,C2,CC,29,0F,8D,C3
1009 DATA
CC,A9,80,8D,C1,CC,AD,C2,CC,0A
1010 DATA
0A,0A,0A,0D,C1,CC,0D,C3,CC,8D
1011 DATA
0B,DC,AD,C5,CC,0A,0A,0A,0A,0D
1012 DATA
C6,CC,8D,0A,DC,A9,00,8D,09,DC
1013 DATA
8D,08,DC,78,A9,7F,8D,0D,DC,A9
1014 DATA
80,8D,0E,DC,A9,01,8D,1A,D0,A0
1015 DATA
1B,8C,11,D0,A9,22,8D,12,D0,A9
1016 DATA
D5,A0,CB,8D,14,03,8C,15,03,A9
1017 DATA
8F,A0,CC,8D,18,03,8C,19,03,58
1018 DATA
60,A2,0C,BD,1B,04,9D,E0,03,BD
1019 DATA
1B,D8,9D,ED,03,A9,01,9D,1B,D8
1020 DATA
CA,10,EC,AD,0B,DC,85,02,29,70
1021 DATA
20,86,CC,8D,1C,04,A5,02,20,8A
1022 DATA
CC,8D,1D,04,AD,0A,DC,48,20,86
1023 DATA
CC,8D,1F,04,68,20,8A,CC,8D,20
1024 DATA
04,A9,A0,8D,1B,04,8D,24,04,8D
1025 DATA
27,04,A9,AE,8D,21,04,A5,02,30
1026 DATA
04,A9,41,D0,02,A9,50,29,3F,09
1027 DATA
80,8D,25,04,A9,4D,29,3F,09,80
1028 DATA
8D,26,04,AD,09,DC,48,20,86,CC
1029 DATA
8D,22,04,68,20,8A,CC,8D,23,04
1030 DATA
AD,08,DC,C9,05,B0,04,A9,3A,D0
1031 DATA
02,A9,20,09,80,8D,1E,04,A9,3C
1032 DATA
CD,12,D0,D0,FB,A2,0C,BD,E0,03
1033 DATA
9D,1B,04,BD,ED,03,9D,1B,D8,CA
1034 DATA
10,F1,EE,19,D0,4C,31,EA,C9,30
1035 DATA
90,04,C9,3A,90,01,38,60,4A,4A
1036 DATA
4A,4A,29,0F,09,B0,60,A9,37,85
1037 DATA
01,20,A3,FD,20,15,FD,20,18,E5
1038 DATA
20,A5,CB,A2,FB,9A,4C,74,A4,0D
1039 DATA
05,50,4F,44,41,4A,20,47,4F,44
1040 DATA
5A,49,4E,45,20,28,4E,50,2E,30
1041 DATA 39,3A,31,33,29,3A,00,00
```

Listing 7

```
/* Program w ARexx'ie
zapoznajacy uzytkownika
Cygnus'a z aktualnym czasem */

LF='0A'X

ADDRESS 'rexx_ced'

Time=TIME('H')||':'||TIME('M')//60

TimeString='The current time
is'||LF||CENTER(Time,19)

Okayl TimeString
```

Listing 8

```
************************
*     Przerwania*
*     by K.K./Qrt*
*(c) 1992 Kebab  *
************************
;Tekst zrodlowy dla SEKA Asm.

Start:      move.w
$dff01c,Status
        move.l      $6c,SaveIrq
        move.l      #MyIrq,$6c
        move.l      #Copper,$dff080
        move.w      #0,Count
        move.w      #$8030,$dff09a

Loop: btst  #6,$bfe001
        bne   Loop
        move.l      SaveIrq,$6c
        move.w      Status(pc),d0
        or.w  #$8000,d0
        move.w      d0,$dff09a
        moveq       #0,d0 rts

*--------

;Procedura obslugi przerwania

MyIrq:      movem.l  d0-d7/a0-
a6,-(a7)
        move.w      $dff01e,d0
        and.w #$0010,d0
        bne   CopInt
        move.w      $dff01e,d0
        and.w #$0020,d0
        bne   VBlankInt
        bra   ExitInt

CopInt: move.w     #$000f,$dff180
        move.w      #$0000,$dff180
        move.w      #$0010,$dff09c
        bra   ExitInt

VBlankInt: addq.w      #1,Count
        cmp.w #50,Count
        bne   Go
        move.w      #0,Count
        bchg  #1,$bfe001

Go:     move.w      #$0020,$dff09c
        bra   ExitInt

ExitInt: movem.l   (a7)+,d0-d7/
a0-a6
        move.l      SaveIrq,Jump+2

Jump:jmp    $00000000

*--------
SaveIrq:dc.l      0
Count:  dc.w      0
Status: dc.w      0

*--------
Copper: dc.l      $00960020,$01000000
        dc.l  $01020000,$01040000
        dc.l  $008e2c50,$00902cc1
        dc.l  $00920038,$009400d0

;Wywolanie przerwania
        dc.l  $6021fffe
        dc.l  $009c8010

;Wywolanie przerwania
        dc.l  $a041fffe
        dc.l  $009c8010
        dc.l  $fffffffe
```



Kupon ogłoszeniowy

----
imię i nazwisko

----
adres

treść:

